UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RENAN DE QUEIROZ MAFFEI

Translating sensor measurements into texts for localization and mapping with mobile robots

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Edson Prestes e Silva Jr.
Coadvisor: Prof. Dr. Mariana Luderitz Kolberg

Porto Alegre
April 2017

CIP — CATALOGING-IN-PUBLICATION

Maffei, Renan de Queiroz

Translating sensor measurements into texts for localization and mapping
with mobile robots / Renan de Queiroz Maffei. – Porto Alegre: PPGC da UFRGS, 2017.

163 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2017. Advisor: Edson Prestes e Silva Jr.; Coadvisor: Mariana Luderitz Kolberg.

1. Mobile Robots. 2. Localization. 3. Mapping. 4. Place Recognition. 5. Free-Space Density. 6. n-grams. I. Silva Jr., Edson Prestes e. II. Kolberg, Mariana Luderitz. III. Título.

"It's just a question of reassembling the components in the correct sequence..."

— Alan Moore & Dave Gibbons, "Watchmen", 1985

AGRADECIMENTOS

Em primeiro lugar, agradeço aos meus pais Sérgio e Elenice e à minha irmã Letícia, por tudo que me proporcionaram e seguem proporcionando na vida. Não há outra família como essa.

Agradeço aos meus orientadores professores Edson Prestes e Mariana Kolberg, não apenas pela orientação que foi muito importante para que este trabalho alcançasse o resultado atual, mas também pela excelente convivência (no caso do Edson, por quatro continentes).

Agradeço aos professores Silvia Botelho (FURG), Mário Campos (UFMG) e Paulo Engel (UFRGS), membros da banca na avaliação da proposta de Tese. Seus comentários foram fundamentais para o aprimoramento deste trabalho.

Agradeço ao colega e amigo Vitor A. M. Jorge com o qual trabalhei diariamente durante todo esse tempo, tanto em assuntos de nossas teses de doutorado, quanto em muitos outros trabalhos. Diga-se de passagem, foi uma bela dupla. Da mesma forma, agradeço a Vitor F. Rey que por um bom tempo fez essa dupla virar um ótimo trio. É importante ressaltar que todos os trabalhos apresentados nesta tese foram desenvolvidos em total colaboração com os dois Vitors. Agradeço também aos outros amigos do grupo Phi Robotics Research e ao amigo Rodrigo Oliveira pela companhia diária nesses anos todos.

Por fim, agradeço a Mariane, que além de colaborar neste trabalho, tem sido a melhor companhia que eu poderia esperar um dia ter.

ABSTRACT

Simultaneous Localization and Mapping (SLAM), fundamental for building robots with true autonomy, is one of the most difficult problems in Robotics and consists of estimating the position of a robot that is moving in an unknown environment while incrementally building the map of such environment. Arguably the most crucial requirement to obtain proper localization and mapping is precise place recognition, that is, determining if the robot is at the same place in different occasions just by looking at the observations taken by the robot. Most approaches in literature are good when using highly expressive sensors such as cameras or when the robot is situated in low ambiguous environments. However this is not the case, for instance, using robots equipped only with range-finder sensors in highly ambiguous indoor structured environments. A good SLAM strategy must be able to handle these scenarios, deal with noise and observation errors, and, especially, model the environment and estimate the robot state in an efficient way. Our proposal in this work is to translate sequences of raw laser measurements into an efficient and compact text representation and deal with the place recognition problem using linguistic processing techniques. First, we translate raw sensor measurements into simple observation values computed through a novel observation model based on kernel-density estimation called Free-Space Density (FSD). These values are quantized into significant classes allowing the division of the environment into contiguous regions of homogeneous spatial density, such as corridors and corners. Regions are represented in a compact form by simple words composed of three syllables – the value of spatial density, the size and the variation of orientation of that region. At the end, the chains of words associated to all observations made by the robot compose a text, in which we search for matches of n-grams (i.e. sequences of words), which is a popular technique from shallow linguistic processing. The technique is also successfully applied in some scenarios of long-term operation, where we must deal with semi-static objects (i.e. that can move occasionally, such as doors and furniture). All approaches were evaluated in simulated and real scenarios obtaining good results.

Keywords: Mobile Robots. Localization. Mapping. Place Recognition. Free-Space Density. n-grams.

Traduzindo leituras de sensores em textos para localização e mapeamento de robôs móveis.

RESUMO

Localização e Mapeamento Simultâneos (SLAM), fundamental para robôs dotados de verdadeira autonomia, é um dos problemas mais difíceis na Robótica e consiste em estimar a posição de um robô que está se movendo em um ambiente desconhecido, enquanto incrementalmente constrói-se o mapa de tal ambiente. Provavelmente o requisito mais importante para localização e mapeamento adequados seja um preciso reconhecimento de local, isto é, determinar se um robô estava no mesmo lugar em diferentes ocasiões apenas analizando as observações feitas pelo robô em cada ocasião. A maioria das abordagens da literatura são boas quando se utilizam sensores altamente expressivos, como câmeras, ou quando o robô está situado em ambientes com pouco ambiguidade. No entanto, este não é o caso, por exemplo, quando o robô equipado apenas com sensores de alcance está em ambientes internos estruturados altamente ambíguos. Uma boa estratégia deve ser capaz de lidar com tais ambientes, lidar com ruídos e erros nas observações e, especialmente, ser capaz de modelar o ambiente e estimar o estado do robô de forma eficiente. Nossa proposta consiste em traduzir sequências de medições de laser em uma representação de texto eficiente e compacta, para então lidar com o problema de reconhecimento de local usando técnicas de processamento lingüísticos. Nós traduzimos as medições dos sensores em valores simples computados através de um novo modelo de observação baseado em estimativas de densidade de kernel chamado de Densidade de Espaço Livre (FSD). Estes valores são quantificados permitindo a divisão do ambiente em regiões contíguas de densidade homogênea, como corredores e cantos. Regiões são representadas de forma compacta por simples palavras descrevendo o valor de densidade espacial, o tamanho e a variação da orientação daquela região. No final, as cadeias de palavras compõem um texto, no qual se buscam casamentos de n-gramas (isto é, sequências de palavras). Nossa técnica também é aplicada com sucesso em alguns cenários de operação de longo-prazo, onde devemos lidar com objetos semi-estáticos (i.e. que se movem ocasionalmente, como portas e mobílias). Todas as abordagens foram avaliadas em cenários simulados e reais obtendo-se bons resultados.

Palavras-chave: Robôs móveis, Localização, Mapeamento, Reconhecimento de local, Densidade de Espaço Livre, n-gramas.

LIST OF ABBREVIATIONS AND ACRONYMS

EKF	Extended Kalman Filter
FSD	Free-Space Density
HIMM	Histogramic In-Motion Mapping
ICP	Iterative Closest Point
KDE	Kernel Density Estimate
LIDAR	Light Detection And Ranging
MCL	Monte-Carlo Localization
RBPF	Rao-Blackwellized Particle Filter
SIR	Sampling-Importance-Resampling
SLAM	Simultaneous Localization And Mapping

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\boldsymbol{x}_t$	robot pose $(x, y, \theta)^T$ at instant t. Each component of $\boldsymbol{x}_t$, can be refered by $x(\boldsymbol{x}_t)$, $y(\boldsymbol{x}_t)$ and $\theta(\boldsymbol{x}_t)$.
$\boldsymbol{u}_t$	action that moves the robot from $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t$.
$\boldsymbol{z}_t$	vector of K observations $(\boldsymbol{z}_t^1, \boldsymbol{z}_t^2, \cdots, \boldsymbol{z}_t^K)^T$ made by the robot at instant t. Dependent on the type of sensor used: it may correspond to observations of K landmarks, or to K readings of a range finder sensor. In this case, an observation $\boldsymbol{z}_t^k = (r, \theta)^T$ indicates the measured range r obtained at angle θ.
$\boldsymbol{m}$	map of the environment, i.e. vector $(\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_N)^T$ of all objects (e.g. feature or grid cell) describing the scenario. In occupancy grids, a cell $\boldsymbol{m}_i = (x, y, occ)^T$ is associated to a position (x, y) and an occupancy value occ.
$\mathcal{X}_t$	set $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_M\}$ of particles estimating the robot state at instant t.
$\boldsymbol{p}_t^{[m]} = \langle \boldsymbol{x}, w \rangle$	the m-th particle of $\mathcal{X}_t$ at instant t. $\boldsymbol{p}_t^{[m]}$ is composed of the pose $\boldsymbol{x}$ and the importance weight w.
$\mathcal{G} = \langle \boldsymbol{s}, \mathcal{A} \rangle$	pose graph with nodes given by the state vector $\boldsymbol{s} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)^T$ of all robot poses, and arcs (i.e. edges of directed graph) given by the set $\mathcal{A} = \{\boldsymbol{a}_{01}, \cdots, \boldsymbol{a}_{ij}, \cdots\}$.
$\boldsymbol{a}_{ij} = \langle \boldsymbol{r}, \boldsymbol{\Omega} \rangle$	observed arc between nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, where $\boldsymbol{r}$ is the observed pose[1] of node $\boldsymbol{x}_j$ relatively to node $\boldsymbol{x}_i$, and $\boldsymbol{\Omega}$ is the information matrix describing the uncertainty of the estimate.
$\hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s})$	expected arc between nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, computed based on the current values on the state vector $\boldsymbol{s}$.
$\boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})$	error associated to the arc between nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, given by the difference between $\hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s})$ and $\boldsymbol{r}(\boldsymbol{a}_{ij})$.
$\boldsymbol{J}_{ij} = \left.\frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{s}}\right\|_{\boldsymbol{s}=\breve{\boldsymbol{s}}}$	Jacobian matrix of the error $\boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})$ computed at the state guess $\breve{\boldsymbol{s}}$.

[1]Information obtained after analyzing sensor measurements, e.g. through scan matching.

$\hat{f}_h(x)$	estimation of a probability density function f at point x obtained using a kernel function $K_h(d)$.
$K_h(d)$	circular kernel function with bandwidth h that operates over a point at distance d.
$UK_h(d)$, $GK_h(d)$, $IK_h(d)$	three different kernel profiles with bandwidth h: uniform, Gaussian and inverted, respectively.
Ψ	*soft* free-space density (FSD). It can be obtained in complete or incomplete maps.
$\Psi^\diamond$	*hard* free-space density (FSD). Only obtained in complete maps, otherwise is undefined (UND_Ψ).
$\boldsymbol{o}_t = \langle \Psi, d, \boldsymbol{x}_t \rangle$	observation made at instant t, composed of a FSD value (Ψ) with its associated density class (d), and the current robot pose ($\boldsymbol{x}_t$) given by odometry.
R_i	the i-th contiguous region visited by the robot. $R_i = [\boldsymbol{o}_j, \cdots, \boldsymbol{o}_{j+k}]$ is described by a list of sequential observations of same density class.
O	list of all observations made by the robot. It is equivalent to the concatenation[2] of all regions visited by the robot $O = [R_0 \oplus R_1 \oplus \cdots]$.
$\mathtt{W} = \langle \mathtt{d}, \mathtt{s}, \mathtt{a} \rangle$	word that describes a segment of the robot path with homogeneous FSD. It has three syllables: the density class of all observations in the segment ($\mathtt{d}$), the size of the segment ($\mathtt{s}$), and the angle variation in the segment ($\mathtt{a}$).
$\rho_{\mathtt{s}}(\mathtt{W}_i, \mathtt{W}_j)$	ratio between the sizes of two words, $\mathtt{W}_i$ and $\mathtt{W}_j$, used to compute the matching of words.
$\epsilon_{\mathtt{s}}$	size tolerance to accept a match of two words.
$\epsilon_{\mathtt{a}}$	angle tolerance to accept a match of two words.
$\mathbb{W} = \langle \mathtt{d}, \mathtt{s}, \mathtt{a}, \mathtt{P}, \mathtt{O} \rangle$	word used in the multi-level approach. It is composed by the same three syllables ($\mathtt{d}, \mathtt{s}, \mathtt{a}$) of the original definition of word,

[2]We define the symbol $\oplus$ as the operator to concatenate lists.

	`W`, along with a list of predecessor words (`P`) and the list of observations associated to the word (`O`).
$\mathcal{W}_l$	list of words of the l-th level, $\mathcal{W}_l = [\mathbb{W}_{0,l}, \mathbb{W}_{1,l}, \cdots \mathbb{W}_{n,l}]$, with $\mathbb{W}_{j,l}$ being the j-th word of such level.
$\mathscr{g}$	n-gram match, i.e. a list of n sequential pairs of words that can be matched, $[\langle \mathbb{W}_i, \mathbb{W}_u \rangle, \langle \mathbb{W}_{i+1}, \mathbb{W}_{u+1} \rangle, \cdots, \langle \mathbb{W}_{i+n-1}, \mathbb{W}_{u+n-1} \rangle]$
$\epsilon_{\mathscr{g}}$	minimum number of words required to accept an n-gram match.
$\mathscr{O}$	list of lists of word occurrences, used to speed up the initial search for n-gram matches.
$\mathcal{M}$	list of all matches of n-grams that occur at each instant.
δ	best alignment between two sequences of observations.
`f`	number of frontiers of free-space in the boundaries of a kernel, K_h, centered at the robot position.

CONTENTS

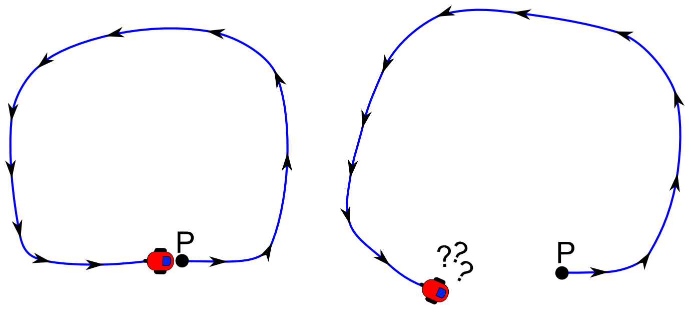
1 INTRODUCTION

In the "*age of the robots*" – label given to the 21st century by many roboticists – we are witnessing intelligent autonomous robots gradually coming out of factories and entering peoples' homes (SICILIANO; KHATIB, 2008). That being said, a minimum requirement for truly autonomous robots is the capability of moving, without supervision, through real-world environments. Therefore their autonomy strongly relies on their ability to maintain the knowledge of their pose, which means localizing themselves when a map of the environment is known a priori, or, otherwise, simultaneously building a map of the environment and self-localizing (SIEGWART; NOURBAKHSH, 2004). The latter problem, called SLAM (Simultaneous Localization and Mapping), is one of the most challenging problems in mobile robotics because it is a kind of *chicken-and-egg* problem: building a map of the environment requires the robot pose to be known, yet, the uncertainty regarding the robot pose increases as the robot moves due to accumulated errors in odometry, and this uncertainty can only be reduced by comparing the robot observations with a map of the environment (THRUN; BURGARD; FOX, 2005).

During the last decades, the Robotics community has deeply studied the problem of simultaneous localization and mapping (DURRANT-WHYTE; BAILEY, 2006). Solutions to such problem propose ways of dealing with sensors uncertainties and properly representing the scenario, while accurately distinguishing ambiguities in the environment. Despite numerous robust approaches presented in the literature – such as based on Kalman filters (SMITH; SELF; CHEESEMAN, 1990; JULIER; UHLMANN, 1997), particle filters (MONTEMERLO et al., 2002; ELIAZAR; PARR, 2004), or graph optimization (LU; MILIOS, 1997a; BOSSE et al., 2003; THRUN; MONTEMERLO, 2006; GRISETTI et al., 2010b) – the area still poses great challenges, specially regarding place recognition, which is crucial for loop closing detection. For example, as illustrated in Figure 1.1(a), a robot that starts at a given point P in an environment and then travels around a large loop that goes back to P, may do not realize, based on dead reckoning, that its current position is actually P. The accumulation of errors arising from integration of odometry readings, discretization of odometry sensors or even wheel slippages lead to significant discrepancy in the pose estimate, as shown in Figure 1.1(b). Therefore, determining if observations made by the robot in different occasions correspond to the same region of the environment is a fundamental task for proper localization and mapping.

An important element of a place recognition strategy is the sensor observation model,

Figure 1.1: Difficulty of loop closing detection. (a) Real robot path in a situation of loop closure. (b) Estimated robot path distorted from the real one due to cumulative odometry errors.



(a) (b)

which depends on the type of sensor used by the robot. For instance, robots equipped with cameras solve place recognition through sum of absolute differences of image patches (MILFORD; WYETH, 2012), comparisons of color histograms of images (WERNER; SITTE; MAIRE, 2012), comparisons of features extracted from images (CUMMINS; NEWMAN, 2008), etc. Robots equipped with range finders, such as sonars or LIDAR [1], solve place recognition by comparing the ranges measured by the sensor, or by matching the endpoints of the sensor beams, among other strategies (BESL; MCKAY, 1992; THRUN; BURGARD; FOX, 2005; SONG; LI, 2012).

Another important element is the world modeling, which must be adapted to the type of the environment. Using sets of lines or planes to represent obstacles may be a good approach if we are working in indoor/structured environments, but not suitable for outdoor terrains (SACK; BURGARD, 2003). In the latter cases, many methods use point clouds to represent the environment. The problem of using point clouds is the low representativeness of the information, that makes difficult both the visualization process and the robot decision process for navigation (BURGARD; HEBERT, 2008). Some world modeling approaches more representative than point clouds are: occupancy grids, which discretize the environment into squared cells through a process that is easy to implement and fast to update, but expensive in terms of memory consumption (HAHNEL et al., 2003a; GRISETTI; STACHNISS; BURGARD, 2007; PRESTES et al., 2003); irregular meshes, or collections of convex polygons, which can represent more accurately the real environment, but are more computationally expensive to maintain than grids; elevation grids and cost maps, that project 3D information into a 2D grid representation (LACROIX et al., 2002); among others. Finally, there are topological maps, which rep-

[1]LIDAR: *Light Detection and Ranging*, i.e. laser range finder

resent the environment using a graph, where nodes are locally contiguous regions of the environment. There are many approaches for node generation, e.g., nodes may be associated to uniform-sized segments of the robot trajectory, or to vertices of a Voronoi diagram extracted from the environment, or even may have a semantic meaning, such as rooms and corridors (WERNER et al., 2009; BEESON; JONG; KUIPERS, 2005; CHOSET; NAGATANI, 2001).

A last fundamental element of a place recognition strategy is how we match sets of observations. A possible approach is to search one-to-one matches of observations (CUMMINS; NEWMAN, 2008), which is appropriate when there are highly distinguishable features in the environment, otherwise it may generate countless false positives. Other possible approach is the sequential match of observations. This makes sense given that a robot that is revisiting a place will generally obtain a sequence of observations that is similar to some other sequence observed in the past. In fact, it has been shown by many methods that analyzing sequential information is a way to disambiguate seemingly indistinguishable regions (MILFORD; WYETH, 2012; WERNER et al., 2009). However, this type of approach strongly depends on a model that is compact and efficient in the management of large sets of data. Additionally, the model should be robust to uncertainties inherent to sensor readings and small changes in the environment, because the chances of finding exact matches of long sequences of measurements are extremely low.

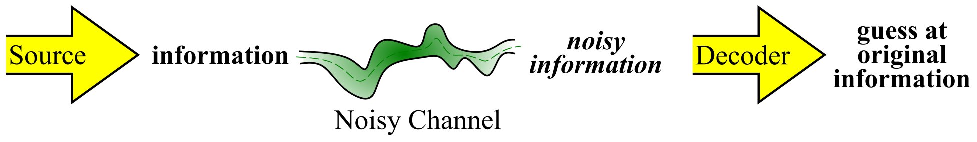
In summary, the key problems associated with robot localization and mapping that we will address in this thesis are

- How to efficiently solve place recognition in indoor structured environments that are highly ambiguous?
- How to model the observations and the environment with a robust and compact representation?
- How to deal with noise and errors in the robot pose estimate and sensor observations, especially in lifelong operation?

1.1 Hypothesis and Goals

As summarized in previous section, to solve the SLAM problem, and more specifically the place recognition problem, we must deal with large amounts of information (i.e. sensor measurements) that are noisy, possibly contaminated with outliers and, in many cases,

Figure 1.2: Noisy channel model for transmission of information. Figure adapted from (JURAFSKY; MARTIN, 2000).



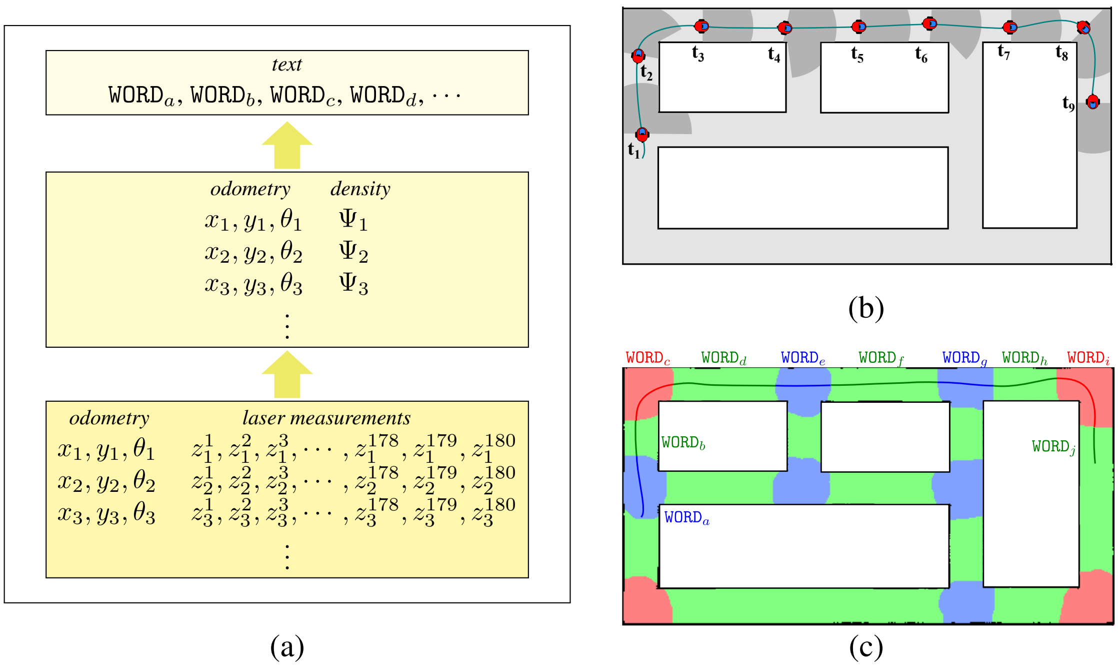
highly ambiguous. We must correct and simplify this information in order to extract a coherent understanding from it (i.e. the map of the environment and the robot localization), which can be done by searching and identifying existing correspondences inside the data. This process resembles what Shannon described, in the seminal paper about information theory (SHANNON, 1948) back in the 1940s, as the metaphor of the "noisy channel" for the transmission of language, illustrated in Figure 1.2. The idea is that the information received in a communication process is a noisy version of the original information, i.e. the information emitted by the source is transformed after passing through a noisy communication channel. Then, we must try to figure out how the channel modified the 'true' information in order to correct it. During decades the problems of correction and prediction of information transmitted according to a language have been the focus of study of different fields in computer science, such as computational linguistics, natural language processing and speech recognition.

We conjecture that the problem of state estimation in robotics can be viewed in the light of linguistic processing. Our idea is that the sequential description of sensor measurements obtained by a robot corresponds to a message describing an information (i.e the environment where the robot is located) that is transmitted through a noisy channel (i.e. contaminated by systematic and non-systematic errors of the robot sensors and actuators). The received information must be analyzed, simplified and corrected to allow the comprehension of the original information. For example, we can view the problem of place recognition as finding matches of very similar strings in the full sequence of transmitted messages. Furthermore, if we define a text description for the sequences of sensor measurements, that is robust and much more compact than the raw readings, than we can efficiently handle the problem with the aid of techniques from linguistic processing.

Our proposal in this work is to translate into a text the observations made by a robot equipped with a laser range finder and to deal with the place recognition problem using techniques from shallow linguistic processing, such as n-grams[2]. The core idea of

[2]An n-gram is an ordered sequence of size n. A gram can be a word (as in our case), a syllable, a letter, etc. n-grams are used for language identification, string matching, among other applications (JURAFSKY; MARTIN, 2000).

Figure 1.3: Core idea of our proposal. (a) Incremental abstraction of raw sensor measurements into a simple text representation. (b) Input: sequence of raw range-finder measurements (dark gray areas in front of the robot). (c) Goal: sequence of words describing different types of regions, such as bifurcations (blue), corridors (green) and corners (red).


text
$\texttt{WORD}_a, \texttt{WORD}_b, \texttt{WORD}_c, \texttt{WORD}_d, \cdots$
odometry density
x_1, y_1, θ_1 Ψ_1
x_2, y_2, θ_2 Ψ_2
x_3, y_3, θ_3 Ψ_3
odometry laser measurements
x_1, y_1, θ_1 $z_1^1, z_1^2, z_1^3, \cdots, z_1^{178}, z_1^{179}, z_1^{180}$
x_2, y_2, θ_2 $z_2^1, z_2^2, z_2^3, \cdots, z_2^{178}, z_2^{179}, z_2^{180}$
x_3, y_3, θ_3 $z_3^1, z_3^2, z_3^3, \cdots, z_3^{178}, z_3^{179}, z_3^{180}$
(a)
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9
(b)
$\texttt{WORD}_c$ $\texttt{WORD}_d$ $\texttt{WORD}_e$ $\texttt{WORD}_f$ $\texttt{WORD}_g$ $\texttt{WORD}_h$ $\texttt{WORD}_i$
$\texttt{WORD}_b$ $\texttt{WORD}_a$ $\texttt{WORD}_j$
(c)

our approach is abstraction, as illustrated in Figure 1.3(a). We start from a sequence of raw measurements, i.e. odometry values (x_i, y_i, θ_i) and laser range-finder readings $(z_i^1, z_i^2, \cdots, z_i^{179}, z_i^{180})$, as exemplified in Figure 1.3(b). The range-finder information is transformed into density values (Ψ_i) using a simple (and efficient) observation model, then we compose more significant information by grouping sequences of similar elementary data. This significant information is translated into words ($\texttt{WORD}_a$, $\texttt{WORD}_b$, $\cdots$), and sequences of words are grouped into a text. Place recognition is made by finding matches of sequences of words, or n-grams. By the end of the process, all the sensors measurements obtained during the robot navigation are converted into a *corpus* that is much smaller and easy to work than the original raw information.

The human readability of the resulting words is low (as will be presented in Chapter 4, the words are composed by three syllables describing free-space density, region length and variation of orientation), however, this is not an issue for the place recognition problem. Nevertheless, later in this thesis, we study a simple way of adding semantic information to the words. For example, the robot path shown in Figure 1.3(c) passes over three types of places: bifurcations (blue), corridors (green) and corners (red). We show that the words generated by our approach serve as a good basis for classification techniques, indicating a great potential for further studies in the area of semantic mapping.

1.2 Current Contributions

The main contributions presented in this work to give support to our conjecture are:

- The translation of raw sensor readings into simplified observation values, that are obtained with a novel observation model called Free-Space Density (FSD). This model computes a single-valued density measure by applying a kernel density estimate (KDE) over the free space surrounding the robot position. The proposed density measure is independent of orientation, what allows an efficient pre-caching step, substantially boosting the computation time of the process. Using the gradient of the densities field, our strategy is able to estimate orientation information that helps to restrict the localization search space.

- An efficient application of the proposed model in the global localization problem using particle filters, where we know the map a priori but do not have initial estimate of the robot pose. We also test combinations of densities obtained by kernels of different sizes and profiles to improve the quality of the acquired information.

- The translation of simplified observation values (described according to the FSD model) into a text. We use the novel observation model as the basis to build words representing regions of the environment. These words are classified based on the density of free space, number of observations and variation in the robot orientation while visiting the corresponding region.

- A place recognition strategy that matches sequences of consecutive words describing the environment. In this case, we consider a robot lost in an unknown environment trying to construct a topological map to localize itself. The analysis of word sequences, i.e. n-grams, is a popular technique used for prediction and correction of information in shallow linguistic processing. We show that our approach for place recognition is much faster than a sequence of Iterative Closest Point (ICP) matches, in experiments performed in real and simulated static scenarios.

- A multi-level extension of the place recognition approach for lifelong operation. In such case, we must deal with environments that contain semi-static objects, such as doors and furniture, which change in a much slower rate than dynamic objects, such as people walking by the robot, and therefore, are harder to properly handle. Every time that a word is built, our approach verifies the possibility of creating alternative words to contemplate the situation where the density variation is caused by a semi-static object. Experiments reveal improvements in relation to the original approach for static scenarios.

- An in-depth analysis from a semantic viewpoint of the proposed FSD-based words. We examine the words associated to different types of places, such as bifurcations, corridors and corners, in simulated and real scenarios (that are widely used in SLAM literature). Our analysis verifies what are the most robust features that can be used to classify each type of place (e.g. positive FSD variations always occur in corners and bifurcations, differently from corridors and rooms). Finally, as a kick-off application of our technique in semantic mapping, we propose a simple decision-tree for word classification based on the results of the analysis, which obtains good results in most of the places of the tested scenarios.

1.3 Organization

This thesis is organized as follows. First, in Chapter 2, we provide the theoretical background of this thesis, reviewing some of the main problems in robot state estimation (e.g. localization, mapping and SLAM), and three techniques used throughout this thesis in assistance to our contributions. We also review kernel density estimation and the n-grams technique used in shallow linguistic processing. In Chapter 3 we detail the Free-Space Density, our novel observation model based on kernel density estimation. In Chapter 4 we describe our approach for building a text from sensor observations, and solving the place recognition problem with techniques from shallow linguistic processing. In Chapter 5 we describe the extension of our place recognition approach for long-term operation. In Chapter 6, we present an analysis of the words generated by our method with a semantic viewpoint. Lastly, in Chapter 7, we draw conclusions about the current work and discuss future directions.

2 THEORETICAL BACKGROUND

This chapter presents a theoretical background detailing techniques used throughout this thesis. First, we address the basic concepts from state estimation in mobile robotics. Our focus in this thesis is the SLAM problem, however we also address mapping and localization individually. Just illustrating, our first contribution is a novel observation model that maps the information surrounding the robot to a single density value. Such technique is tested in the global localization problem. Therefore, we present a review of the three problems (localization, mapping and SLAM), detailing methods that serve as basis for this work.

We also present a review on kernel density estimation and its application to image processing, which serves as basis for our novel observation model, presented in Chapter 3, that computes the free-space density in the environment. Later, we introduce a brief background on linguistic processing and the n-grams technique, which is a fundamental point of our place recognition strategy presented in Chapter 4 and extended in Chapter 5.

2.1 State Estimation in Mobile Robotics

An autonomous mobile robot should be able to move in an environment to fulfill its tasks, which demands that the robot knows its location along with the location of nearby obstacles. In realistic scenarios such information is not directly available, and the robot must use its sensors, which carry only partial and noisy information about the scenario state (THRUN; BURGARD; FOX, 2005).

In summary, there are four variables in the context of state estimation in mobile robotics:

- $\boldsymbol{x}_t$ is the robot pose $(x, y, \theta)^T$ at instant t.[1] The trajectory followed by the robot is represented by $\boldsymbol{x}_{0:t} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t\}$, where $\boldsymbol{x}_0$ is the initial robot pose.

- $\boldsymbol{m}_i$ is the position of an object i in the environment. The map of the environment is given by the vector of all objects, i.e. $\boldsymbol{m} = (\boldsymbol{m}_1, \boldsymbol{m}_2, \cdots, \boldsymbol{m}_N)^T$.

- $\boldsymbol{u}_t$ is the control vector applied at instant $t-1$ that takes the robot to the pose $\boldsymbol{x}_t$ at instant t. It is generally given by the odometry measurement between the instants $t-1$ and t. The history of control commands is represented by $\boldsymbol{u}_{1:t} =$

[1]We use the following functional notation to refer to components of variables represented by tuples or vectors: *component*(**variable**). For instance, the components of $\boldsymbol{x}_t$ are referred as $x(\boldsymbol{x}_t)$, $y(\boldsymbol{x}_t)$ and $\theta(\boldsymbol{x}_t)$.

$\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_t\}$.

- $\boldsymbol{z}_t^i$ is the i-th observation made by the robot at instant t. $\boldsymbol{z}_t = (\boldsymbol{z}_t^1, \boldsymbol{z}_t^2 \cdots, \boldsymbol{z}_t^K)^T$ is the vector of all observations made by the robot at instant t, while $\boldsymbol{z}_{1:t} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_t\}$ represents the history of all observations.

The set of controls $\boldsymbol{u}_{1:t}$ and observations $\boldsymbol{z}_{1:t}$ are always known, given that they represent proprioceptive[2] and exteroceptive[3] sensors information, respectively. The robot state $\boldsymbol{x}_{0:t}$ and the map $\boldsymbol{m}$ may be known or unknown. In fact, as shown in Figure 2.1, there are three main state estimation problems:

- *Localization*: estimating the robot pose when the map is known;
- *Mapping*: estimating the map when the robot pose is known;
- *SLAM (Simultaneous Localization and Mapping)*: estimating both the robot pose and the map.

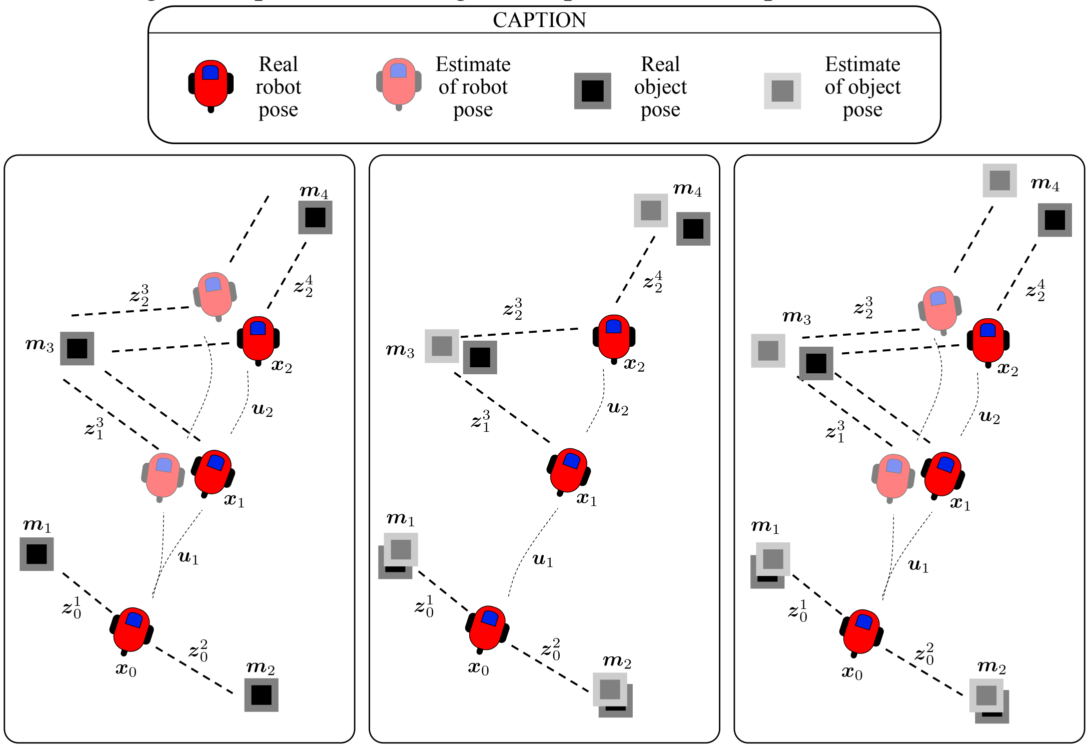
Localization is the problem of determining the pose of a robot in relation to a map of the environment known a priori. It can be seen as a problem of coordinate transformation, i.e. establishing correspondence between a global coordinate system, in which all objects of interests are expressed, and the local coordinate system of the robot (THRUN; BURGARD; FOX, 2005). There are two types of localization problems considering the knowledge available at the start of the process and at run-time. In **local localization**, or *position tracking*, the initial robot pose is known, therefore the localization method must only accommodate the uncertainty in robot motion, generally using a Gaussian distribution. The other type, **global localization**, is more complex and has two variations: the *wake-up robot problem*, where the initial robot pose is unknown, thus it requires multimodal probability distributions to model the localization uncertainty; and the *kidnapped robot problem*, where a well localized robot is teleported to arbitrary locations. This variation of the problem measures the ability of the localization strategy to recover from failures, since when a kidnapping occurs the robot thinks that knows where it is but in fact it does not.

The global localization graphical model, which shows the sequences of variables and the causal relationships between them, is presented in Figure 2.2(a). In the localization

[2]*Proprioception*: measurements of movements relative to an internal frame of reference of the robot. Example: odometry measurements. (MURPHY, 2000)

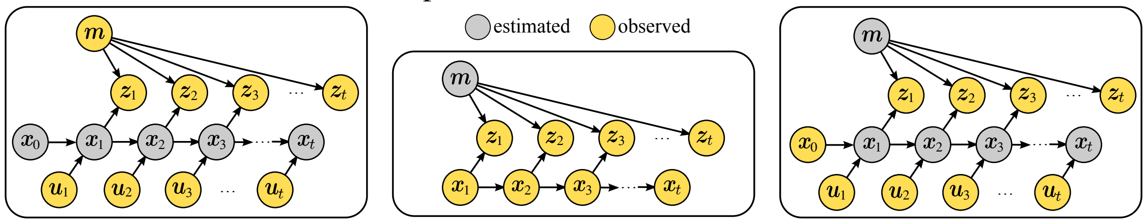
[3]*Exteroception*: measurements of the layout of the environment and objects relative to the robot's frame of reference. Example: laser range finder measurements. (MURPHY, 2000)

Figure 2.1: State estimation problems in mobile robotics. (a) Estimating the robot pose. (b) Estimating the map. (c) Estimating the map and the robot pose at the same time.



(a) Localization (b) Mapping (c) SLAM

Figure 2.2: Graphical models of Localization, Mapping and SLAM. Robot pose $\boldsymbol{x}_{0:t}$ and/or the map $\boldsymbol{m}$ must be estimated based on the observed states, which include controls $\boldsymbol{u}_{1:t}$ and measurements $\boldsymbol{z}_{1:t}$. Adapted from (THRUN; LEONARD, 2008).


estimated
observed
m
z_1 z_2 z_3 ··· z_t
x_0 x_1 x_2 x_3 ··· x_t
u_1 u_2 u_3 ··· u_t

(a) Localization (b) Mapping (c) SLAM

problem, the sequences of robot poses $\boldsymbol{x}_{0:t}$ must be estimated from the controls $\boldsymbol{u}_{1:t}$, the measurements $\boldsymbol{z}_{1:t}$ and the map $\boldsymbol{m}$. (In position tracking $\boldsymbol{x}_0$ is also an observable variable). A traditional way to solve the localization problem is through Markov localization, by applying the Bayes Filter over the belief, $p(\boldsymbol{x}_t|\boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{m})$, about the robot pose at instant t. Such approach follows the Markov assumption – or complete state assumption – which states that the previous belief is sufficient to represent the past history of the robot. In other words, the Bayes filter computes the belief at time t based on the belief at time $t-1$. It is noteworthy that the Markov assumption in robotics is only an approximation, due to unmodeled dynamics and other inaccuracies. Yet, as shown in the literature, Bayes filters are robust in practice (THRUN; BURGARD; FOX, 2005).

Among the most popular implementations of the Markov localization are the Kalman filters (LEONARD; DURRANT-WHYTE, 1991), which assume Gaussian noise and linear motion models; grid-based filters (BURGARD et al., 1998b), which discretize the state space into a grid to deal with multi-modal distributions; and particle filters (DELLAERT et al., 1999), which maintain a weighted set of samples (particles) to represent the posterior distribution. The latter approach, also called Monte Carlo Localization (MCL), proved to be one of the most powerful implementations of robot localization, and is the one used in this work.

Mapping (with known poses) is the problem of estimating the belief about the map, $p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t})$, that is, an accurate representation of the environment based on the observations made by the robot during a known trajectory. Figure 2.2(b) presents the graphical model of the mapping problem. Differently from the localization problem, the map $\boldsymbol{m}$ is the unknown variable, while the sequences of robot poses $\boldsymbol{x}_{1:t}$ and sensor measurements $\boldsymbol{z}_{1:t}$ are known. The set of controls $\boldsymbol{u}_{1:t}$ are also known, because they are measures of proprioceptive sensors, however they are irrelevant for the mapping process given that the robot state is known. We also discard $\boldsymbol{x}_0$ from the mapping process because, by convention, the sensor observations are made after the robot motion, therefore there are no observations made in the initial state.

Different types of maps are used in mobile robotics but generally they are divided in two main approaches: based on features or on locations. In **feature-based maps**, feature extractors are applied over the observations (e.g. scan readings and camera images) to obtain features, such as corners (HARRIS; STEPHENS, 1988; ROSTEN; PORTER; DRUMMOND, 2010), lines (SACK; BURGARD, 2003; CHOI; LEE; OH, 2008; JORGE, 2012), or other visual features (LOWE, 2004; BAY; TUYTELAARS; GOOL, 2006). On the other hand, **location-based maps** discretize the environment into specific locations. In this case, each component, $\boldsymbol{m}_i$, of the map is associated to a region of the environment. Occupancy grid maps are the most popular location-based maps (MORAVEC; ELFES, 1985; HAHNEL et al., 2003b; ELIAZAR; PARR, 2004). They regularly divide the environment into cells, i.e. square regions, where each cell, $\boldsymbol{m_i} = (x, y, occ)^T$, is defined by a position in space (x, y) and an occupancy value (occ). An issue of occupancy grids is that good precision for large environments may require excessive storage space. However, they are easy to maintain, have constant access time, do not rely on any feature extractors and can model not only obstacles, but also free space and unknown regions.

The map of the environment can be considered a set of independent components, which allows the mapping problem to be broken into a collection of separate problems. This assumption neglects dependencies among neighboring objects, nevertheless it makes the map estimation easier by considering the posterior over maps as the product of individual probabilities (THRUN; BURGARD; FOX, 2005)

$$p(\boldsymbol{m} \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}) = \prod_{i=1}^{N} p(\boldsymbol{m}_i \mid \boldsymbol{x}_{1:t}, \boldsymbol{z}_{1:t}) \tag{2.1}$$

In this thesis, we use the HIMM grid strategy as basis for our methods, which is an efficient variation of occupancy grids (BORENSTEIN; KOREN, 1991).

Finally, **Simultaneous Localization And Mapping (SLAM)**, arguably the hardest state estimation problem in mobile robotics, consists in estimating both the robot state $\boldsymbol{x}_t$ and the map $\boldsymbol{m}$ based on the measurements $\boldsymbol{z}_{1:t}$ and controls $\boldsymbol{u}_{1:t}$ made by the robot, as shown by the graphical model in Figure 2.2(c). By convention, the initial robot pose $\boldsymbol{x}_0$ is known[4], otherwise we do not have an initial place to start building the map.

From a probabilistic perspective there are two different forms of SLAM: online SLAM and full SLAM. **Online SLAM** consists in estimating the belief $p(\boldsymbol{x}_t, \boldsymbol{m} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, \boldsymbol{x}_0)$, i.e. the posterior probability on the current robot state $\boldsymbol{x}_t$ and the map $\boldsymbol{m}$. **Full SLAM** estimates the belief $p(\boldsymbol{x}_{1:t}, \boldsymbol{m} \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, \boldsymbol{x}_0)$, i.e. the posterior probability over the entire path traversed by the robot $\boldsymbol{x}_{1:t}$, instead of just the current robot pose, along with the map $\boldsymbol{m}$.

Online SLAM methods implement the Bayes Filter, that is, they perform successive steps of *prediction* and *correction*, respectively considering the control $\boldsymbol{u}_t$ and the observations $\boldsymbol{z}_t$. The earliest online SLAM approaches are based on Extended Kalman Filters (EKF) (SMITH; SELF; CHEESEMAN, 1990; JULIER; UHLMANN, 1997), which linearize motion and observation models (typically non-linear) to estimate the robot state and a feature-based map with Kalman filters. The main drawbacks of such approaches is the quadratical growth in computation cost and memory in terms of the number of features in the map; and the difficulties in dealing with non-linearities in motion and observation models.

More recently, an approach that became very popular was the Rao-Blackwellized particle filter (RBPF) (MONTEMERLO et al., 2002; HAHNEL et al., 2003a; ELIAZAR; PARR, 2004; GRISETTI; STACHNISS; BURGARD, 2007; STACHNISS; GRISETTI;

[4]$\boldsymbol{x}_0$ is usually considered as $(0, 0, 0)^T$, but any other value is equally valid.

BURGARD, 2005; MAFFEI et al., 2013), where each particle represents a hypothesis about the robot trajectory and is associated to one map. RBPFs are able to represent multi-modal distributions, do not require precise solutions for the data association problem and are probably the easiest algorithm to implement. Notwithstanding, they can be too costly or simply diverge when operating in large environments, where the required number of particles may be substantially high. Even though each particle describes the full robot trajectory, RBPFs are not full SLAM approaches, in which all data is processed at the same time. RBPFs follow the Markov assumption, i.e. they estimate the current state of the robot based on one previous state, what makes them incremental and unable to correct previous estimates (THRUN; LEONARD, 2008). In fact, high risk of divergence is an intrinsic issue of online SLAM approaches, because once the state estimate largely deviates from the true solution, the recovery becomes impossible.

That said, full SLAM approaches are now the most popular SLAM form in the literature and generally rely on very powerful graph optimization techniques that operate on the complete set of observations (LU; MILIOS, 1997a; THRUN; MONTEMERLO, 2006; GRISETTI et al., 2010b). These methods have a high cost associated, which was a critical problem in the past, however, in recent years, this constraint has been minimized due to the advances in graph optimization solutions and computational power.

In next sections we describe some state estimation techniques in details. First, we explain the Monte Carlo Localization technique, used in a case study presented in Chapter 3. Next, we explain the HIMM technique used in the mapping problem, which is used as first step for computing our novel observation model, the free-space density, also presented in Chapter 3. At last, we discuss the graph-based SLAM approach in more details, given that one of the main works in this thesis is a place recognition strategy applied in a graph-based SLAM approach, presented in Chapters 4 and 5.

2.1.1 Monte Carlo Localization

The Monte Carlo Localization (MCL) (DELLAERT et al., 1999) is a nonparametric implementation of the Bayes filter, which approximates the posterior $p(\boldsymbol{x}_t|\boldsymbol{u}_{1:t}, \boldsymbol{z}_{1:t}, \boldsymbol{m})$ using a set of M particles,

$$\mathcal{X}_t = \{\boldsymbol{p}_t^{[1]}, \boldsymbol{p}_t^{[2]}, \cdots, \boldsymbol{p}_t^{[M]}\}, \tag{2.2}$$

where each particle $\boldsymbol{p}_t^{[m]} = \langle \boldsymbol{x}, w \rangle$ has an importance weight w associated to its pose $\boldsymbol{x}$ at time t.

Algorithm 2.1 presents the MCL algorithm, which estimates the particle set $\mathcal{X}_t$ recursively from the set $\mathcal{X}_{t-1}$ through a Sampling-Importance-Resampling (SIR) process. The first step of the algorithm is the **sampling** (lines 3-4). In this step, each particle is propagated according to the control $\boldsymbol{u}_t$, which is made by applying the motion model $p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$ over the previous pose of the particle, i.e $\boldsymbol{x}(\boldsymbol{p}_{t-1}^{[m]})$.

Algorithm 2.1: Monte Carlo Localization

```
   Input: $\mathcal{X}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t, \boldsymbol{m}$
   Output: $\mathcal{X}_t$
1  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2  for $m$ in $1...M$ do
3      $\boldsymbol{x}_{t-1} = \boldsymbol{x}(\boldsymbol{p}_{t-1}^{[m]})$
4      sample $\boldsymbol{x} \propto p(\boldsymbol{x}_t \mid \boldsymbol{u}_t, \boldsymbol{x}_{t-1})$
5      $w = p(\boldsymbol{z}_t \mid \boldsymbol{x}, \boldsymbol{m})$
6      $\boldsymbol{p}_t^{[m]} = \langle \boldsymbol{x}, w \rangle$
7      $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t \cup \{\boldsymbol{p}_t^{[m]}\}$
8  for $m$ in $1...M$ do
9      draw $\boldsymbol{p}_t^{[i]}$ from $\bar{\mathcal{X}}_t$ with probability $\propto w(\boldsymbol{p}_t^{[i]})$
10     $\mathcal{X}_t = \mathcal{X}_t \cup \{\langle \boldsymbol{x}_t^{[i]}, 1/M \rangle\}$
11 return $\mathcal{X}_t$
```

The second step of the algorithm is the **importance weighting** (line 5), in which an individual weight is assigned to each particle. This weight is obtained using the observation model, by comparing the real sensor measurements to the particle's estimated sensor measurements. The idea is to compute the similarity between the target distribution, $p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, \boldsymbol{m})$, and the proposal distribution, $p(\boldsymbol{x}_t \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}, \boldsymbol{m})$ (the one obtained after sampling). The propagated and weighted particles compose a temporary particle set $\bar{\mathcal{X}}_t$ (lines 6-7).

The third and final step of the MCL algorithm is the **resampling** (lines 9-10). The method randomly selects with replacement the same quantity M of particles from $\bar{\mathcal{X}}_t$ and add them to a new set $\mathcal{X}_t$. A simple technique that can be chosen to draw samples is the wheel-roulette algorithm, commonly used in genetic algorithms, that sets the probability of selecting each particle $\boldsymbol{p}_t^{[m]}$ in proportion to its weight $w(\boldsymbol{p}_t^{[m]})$. Resampling is fundamental to a particle filter because it is the responsible for approximating the particles distribution to the true posterior. In general, during the resampling, particles with higher weights will be more likely to be replicated than those with lower weights (those tend to be discarded), which makes the filter to converge.

Figure 2.3: Example of the sampling, importance weighting and resampling in a particle filter. (a) Difference between proposal and target distributions. (b) Sampling. (c) Weighting. (d) Resampling. (e) Comparison of the resulting distribution with the proposal and target distributions. Adapted from (MONTEMERLO; THRUN, 2007).

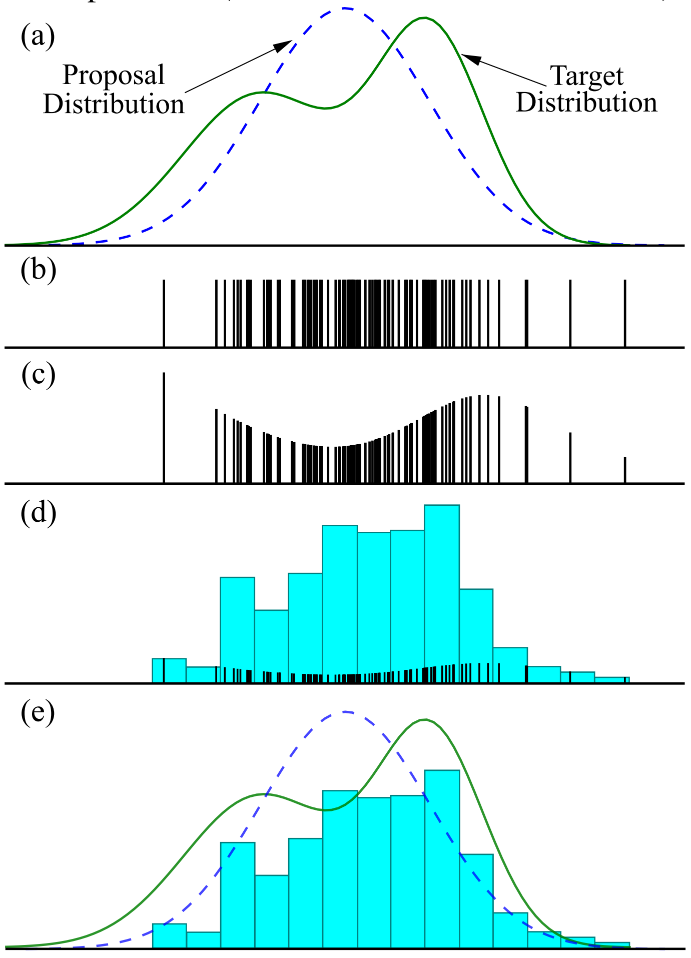


Figure 2.3 illustrates the importance of the resampling step in the MCL. In (a), the target distribution (green solid line) is a multi-modal distribution, while the proposal distribution (blue dashed line) is a simple Gaussian. Since the sampling step, in (b), is performed according to the proposal distribution, the resulting samples (black vertical lines) must be weighted to approximate them to the target distribution. The weighting is performed, in (c), and samples of regions where the proposal distribution is underestimated in relation to the target distribution receive higher weights (taller black lines) than regions where the proposal distribution is overestimated. In the resampling step, (d), the probability of drawing particles from a given region of the distribution (i.e. bucket of the histogram of weights, in blue) is proportional to the summation of the particles weights of such region (i.e. the height of the corresponding bucket). In the end, as shown in (e),

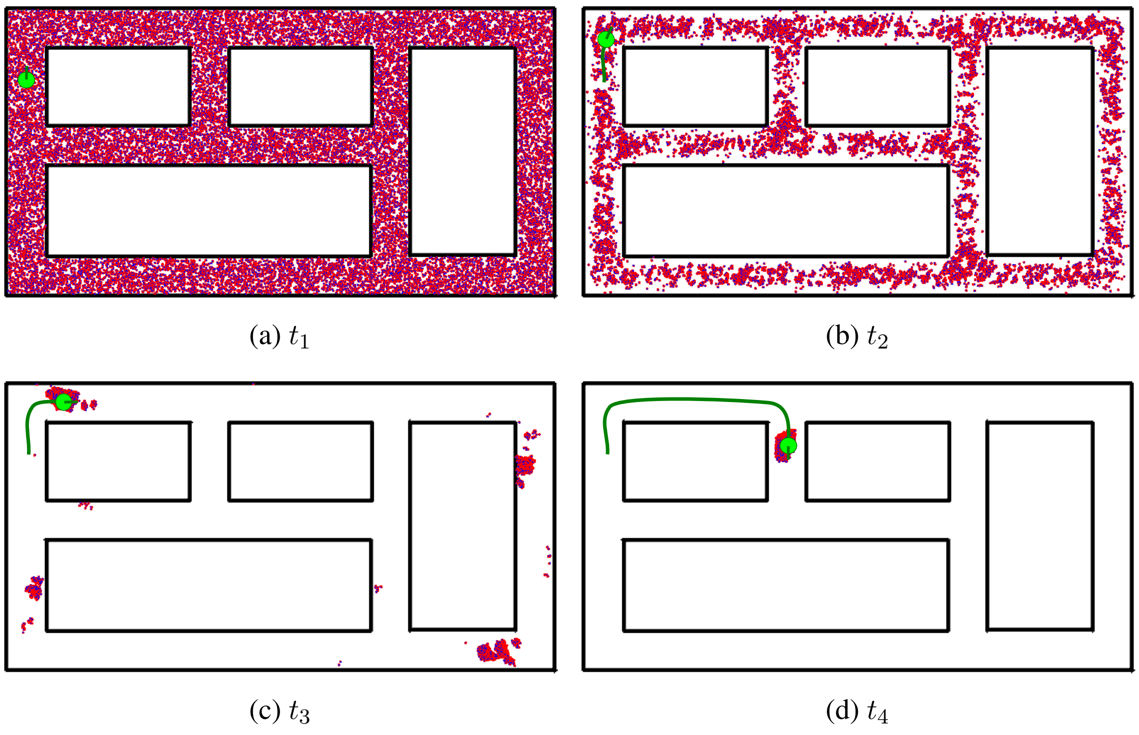
the resulting distribution (blue histogram) must approximate the target distribution (green line). In fact, when we increase the number of particles the filter better approximates to the target distribution.

At last, we should note that Algorithm 2.1 describes the localization process during a single step of the robot trajectory, that is, the movement from instant $t-1$ to t. A complete result of MCL is depicted in Figure 2.4. Initially, in (a), particles are generated in all free space because the robot can be anywhere in the beginning of the process. As the robot moves, the uncertainty decreases. Yet, after only a small displacement, shown in (b), the robot does not have a clue about its location yet. In (c), the robot turns to the right in a corner and the uncertainty drops significantly, because there are only four corners in the environment. Finally, in (d), the robot turns again to the right and the last ambiguities are resolved.

2.1.2 Histogramic In-Motion Mapping (HIMM)

Histogramic In-Motion Mapping (HIMM) is an efficient mapping method that uses a two-dimensional Cartesian histogram grid for obstacle representation measured by range

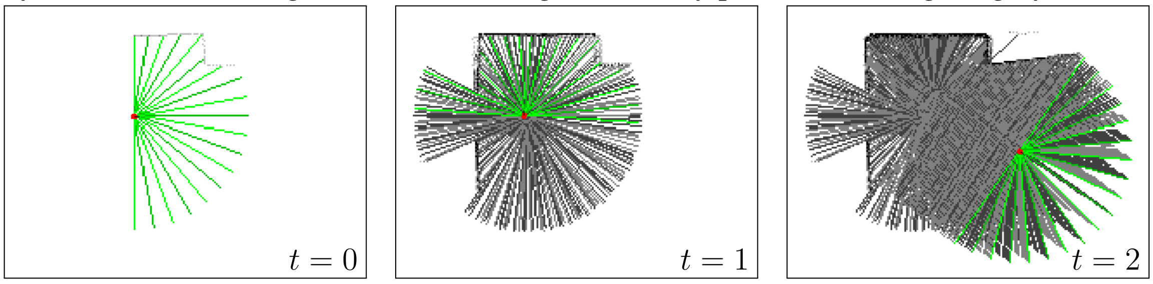
Figure 2.4: Example of particle filter convergence in Monte Carlo Localization, with particles shown in pink, and the robot and its trajectory shown in green. The particles converge when the robot moves enough to resolve the ambiguities in the environment.


(a) t_1 (b) t_2 (c) t_3 (d) t_4

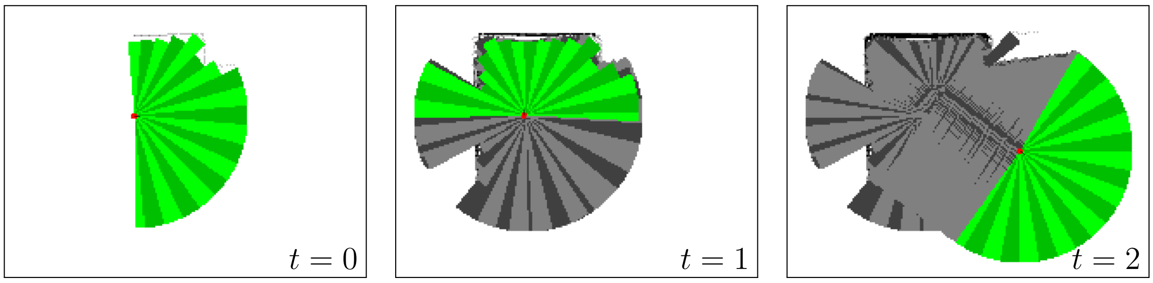
finder sensors (BORENSTEIN; KOREN, 1991). Different from early occupancy grid methods (MORAVEC; ELFES, 1985), which express occupancy probabilities with likelihoods and odds obtained using Bayes rule, the HIMM method divides the occupancy probability space into few possible integer values and updates the occupancy of cells along the acoustic axis of the sensor through simple increments and decrements.

One way to implement HIMM is through the projection, via ray-casting, of each individual sensor reading, $z_t^k = (r, \theta)^T$, over the grid to find the cells to be updated. The problem of this approach is that sampling problems lead to "holes" (i.e. cells with unknown occupancy value) in the middle of the free-space region, as illustrated in Figure 2.5(a). This can be problematic for applications like kernel density estimation, which is the case of the novel observation model that will be presented in Chapter 3. A way to circumvent this problem, although possibly more costly, is to compute the occupancy of all grid cells surrounding the robot inside the perceptual field of the sensor, as shown in Figure 2.5(b).

Figure 2.5: Computing HIMM at each grid cell in the sensor's field-of-view instead of using ray-casting is required to avoid holes in the free-space region. Grid cells updated by current scan readings are illustrated in green, and by past scan readings in gray.



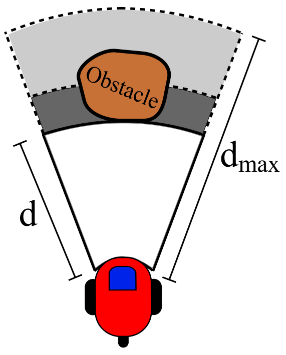
(a) HIMM using ray-casting.



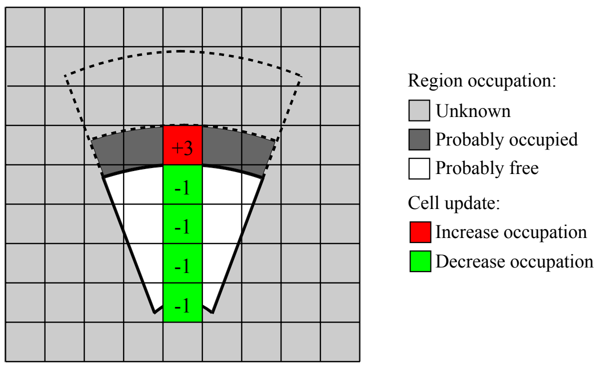
(b) HIMM computed at each grid cell.

Figure 2.6 and Algorithm 2.2 present the HIMM sensor model and HIMM updating strategy, respectively. The minimum occupancy value of a cell is 0, while the maximum is 15, totalizing a range of 16 values (all those values were empirically determined by Borenstein and Koren). At the beginning, all cells are initialized with the minimum value (0), if we consider a priori that the map is empty, or with a middle value (8), representing

Figure 2.6: HIMM model. (a) Robot detecting an obstacle at distance d. (b) Update of grid cells. Only the cells in the axis of the sensor are updated. The occupation of the green cells in free region (white) is diminished and of the red cells in occupied region (dark gray) is augmented. Adapted from (MURPHY, 2000).



(a)



(b)

the initial lack of knowledge about obstacles. At each step of the robot navigation, we first translate[5] the current robot pose to the corresponding grid cell $\boldsymbol{m}_{robot}$ (line 1 of Algorithm 2.2).

Next, we check all cells in the map, $\boldsymbol{m}$, that are inside the perceptual field of $\boldsymbol{z}_t$ (lines 3-4). In practice, if we consider that the maximum sensor range is defined by r_{max} (in meters), which corresponds to d_{max} in terms of number of cells, then we can just check the cells disting at most d_{max} from $\boldsymbol{m}_{robot}$. For each cell, $\boldsymbol{m}_i$, we compute the difference, ϕ_i, between the angle of the vector connecting $\boldsymbol{m}_{robot}$ to $\boldsymbol{m}_i$ and the robot orientation, $\theta(\boldsymbol{x}_t)$ (line 5). We use this angle ϕ_i to find which sensor beam, $\boldsymbol{z}_t^k$, is used to update the cell $\boldsymbol{m}_i$ (line 6). Finally, we compare the distance d_i, between the positions of cells $\boldsymbol{m}_i$ and $\boldsymbol{m}_{robot}$ (line 7), and the distance d, corresponding to the sensor beam measurement $\boldsymbol{z}_t^k$ (line 8). If the cell is around the distance measured by the sensor – i.e. $||d_i - d|| < \epsilon_d$, where ϵ_d is a small distance tolerance – then it belongs to a region probably occupied by obstacles. If the cell is closer to the robot than the distance measured by the sensor, i.e. $d_i < d$, then it belongs to a free region. The update of the cell occupancy is done as follows (lines 9-12): occupied cells are incremented of 3, and free cells are decreased of 1 (totaling at maximum 15 and at minimum 0).

An example of map produced by HIMM is shown in Figure 2.7. As the robot moves, the method consolidates the certainty about what is free space and what are obstacles. Regions that are barely observed, such as frontiers of the sensor's scan, tend to be less

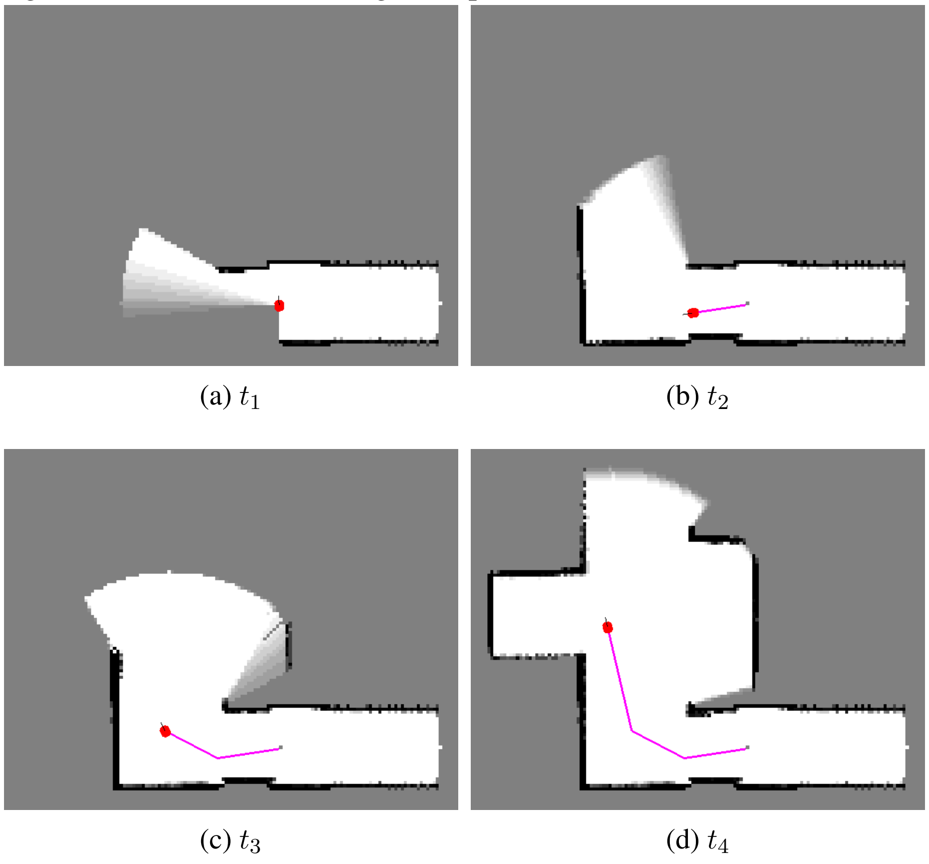
[5]In the algorithm we use two functions, $pose2grid$ and $dist2grid$, to, respectively, represent the translation of a pose coordinate to a cell coordinate, and a distance value in meters to a distance value in cells.

Algorithm 2.2: HIMM

```
Input: $\boldsymbol{x}_t, \boldsymbol{z}_t, \boldsymbol{m}, r_{max}$
Output: $\boldsymbol{m}$
1  $\boldsymbol{m}_{robot} = pose2grid(\boldsymbol{x}_t)$
2  $d_{max} = dist2grid(r_{max})$
3  for each cell $\boldsymbol{m}_i$ in $\boldsymbol{m}$ do
4      if $\boldsymbol{m}_i$ in perceptual field of $\boldsymbol{z}_t$ then
5          $\phi_i = atan2(y(\boldsymbol{m}_i) - y(\boldsymbol{m}_{robot}), x(\boldsymbol{m}_i) - x(\boldsymbol{m}_{robot})) - \theta(\boldsymbol{x}_t)$
6          $k = \arg\min_j |\phi_i - \theta(\boldsymbol{z}_t^j)|$
7          $d_i = \sqrt{(x(\boldsymbol{m}_i) - x(\boldsymbol{m}_{robot}))^2 + (y(\boldsymbol{m}_i) - y(\boldsymbol{m}_{robot}))^2}$
8          $d = dist2grid(r(\boldsymbol{z}_t^k))$
9          if $||d_i - d|| < \epsilon_d \wedge d_i < d_{max}$ then
10             $occ(\boldsymbol{m}_i) = min(15, occ(\boldsymbol{m}_i) + 3)$
11         else if $d_i < d - \epsilon_d$ then
12             $occ(\boldsymbol{m}_i) = max(0, occ(\boldsymbol{m}_i) - 1)$
13 return $\boldsymbol{m}$
```

certain.

Figure 2.7: Example of mapping using HIMM. In this visualization, the brighter the cells are, the higher is the evidence of being free space (*vice versa* for obstacles).



(a) t_1 (b) t_2 (c) t_3 (d) t_4

2.1.3 Graph-based SLAM

Approaches based on graph optimization, which dominate the state-of-the-art of the SLAM literature, solve the problem through nonlinear sparse optimization of a graph of robot poses (LU; MILIOS, 1997a; THRUN; MONTEMERLO, 2006; GRISETTI et al., 2010b). In pose graphs (GRISETTI et al., 2010b), nodes represent robot poses and arcs represent rigid transformations between them. Other methods (THRUN; MONTEMERLO, 2006) also represent features or landmarks as nodes, and corresponding bearing measurements as arcs. In this thesis we work with pose graphs.

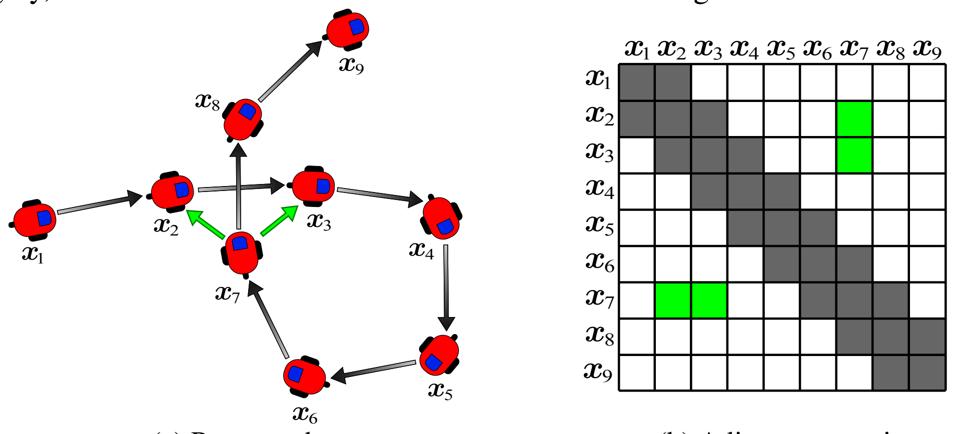
A SLAM pose graph $\mathcal{G} = \langle \boldsymbol{s}, \mathcal{A} \rangle$ is composed of:

- a state vector $\boldsymbol{s} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_i, \cdots, \boldsymbol{x}_t)^T$ representing the nodes of the graph, where each robot pose $\boldsymbol{x}_i$ is a node;

- a set of arcs (i.e. edges of a directed graph) $\mathcal{A} = \{\boldsymbol{a}_{01}, \cdots, \boldsymbol{a}_{ij}, \cdots\}$, where $\boldsymbol{a}_{ij} = \langle \boldsymbol{r}, \boldsymbol{\Omega} \rangle$ is the arc connecting the node $\boldsymbol{x}_i$ to the node $\boldsymbol{x}_j$. $\boldsymbol{a}_{ij}$ is defined by $\boldsymbol{r}$, the observed relative pose of $\boldsymbol{x}_j$ regarding $\boldsymbol{x}_i$, and $\boldsymbol{\Omega}$, the information matrix[6] representing the uncertainty of such estimate.

Figure 2.8 shows a pose graph construction from a sequence of robot poses. As we can see, there are two types of graph arcs: those arising from odometry-based constraints, which connect every consecutive pair of robot poses $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i-1}$ (gray arcs); and those obtained through matchings of observations, which happen when the robot returns to a visited region (green arcs).

Figure 2.8: Example of pose graph construction. Odometry-based constraints are shown in gray, while observation-based constraints are shown in green.



	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$	$\boldsymbol{x}_6$	$\boldsymbol{x}_7$	$\boldsymbol{x}_8$	$\boldsymbol{x}_9$
$\boldsymbol{x}_1$	▨	▨							
$\boldsymbol{x}_2$	▨	▨	▨				🟩		
$\boldsymbol{x}_3$		▨	▨	▨			🟩		
$\boldsymbol{x}_4$			▨	▨	▨				
$\boldsymbol{x}_5$				▨	▨	▨			
$\boldsymbol{x}_6$					▨	▨	▨		
$\boldsymbol{x}_7$		🟩	🟩			▨	▨	▨	
$\boldsymbol{x}_8$							▨	▨	▨
$\boldsymbol{x}_9$								▨	▨

(a) Pose graph (b) Adjacency matrix

[6]The information matrix ($\boldsymbol{\Omega}$) corresponds to the inverse of the covariance matrix ($\boldsymbol{\Sigma}^{-1}$).

Building the graph is considered the **front-end** of graph-based SLAM. In order to find the correct associations between nodes the front-end must be able to detect loop closures, i.e. to solve the place recognition problem. Since the beginning of the 1990s, numerous strategies for place recognition using laser scans have been proposed, such as aligning points to line segments (COX, 1991), matching points through the iterative closest point (ICP) algorithm (BESL; MCKAY, 1992; LU; MILIOS, 1997b), computing a correlative scan matching (OLSON, 2009), extracting local descriptor from scan readings and matching them (GRANSTROM et al., 2009; TIPALDI; ARRAS, 2010), etc. Camera-based front-ends also have been widely proposed in literature and many of them are based on matching of image descriptors, such as SIFT (LOWE, 2004), SURF (BAY; TUYTELAARS; GOOL, 2006), or ORB (RUBLEE et al., 2011; MUR-ARTAL; MONTIEL; TARDÓS, 2015), while others match image patches and bag-of-words (CUMMINS; NEWMAN, 2008; MILFORD; WYETH, 2012).

Nevertheless, just building the correct associations in the front-end is not enough because, since the robot state is initially defined based on odometry, there are errors between odometry-based constraints and observation-based constraints, as shown in Figure 2.9. The correction of such errors is made through an optimization phase, which is referred as the **back-end** of graph-based SLAM. It is worth noting that if we consider only the odometry-based constraints, looking back at Figure 2.8(b), the adjacency matrix would be a tridiagonal matrix[7]. Thus, the observation-based constraints are the responsible for populating the remaining of the matrix, and fortunately, in most cases in SLAM, the resulting matrix is sparse, which allows an efficient optimization step.

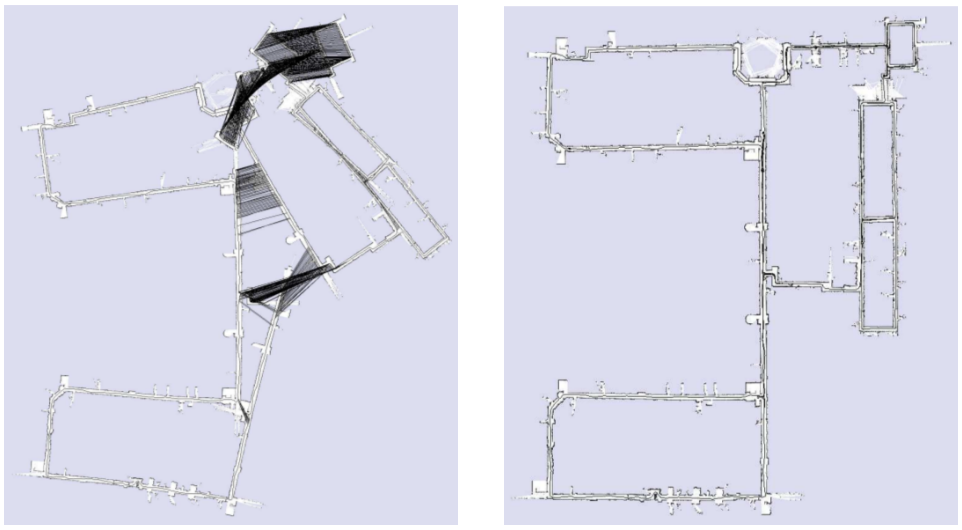
Several types of approaches can be used as graph-based SLAM back-end, such as Gauss-Newton, Levenberg-Marquardt, stochastic gradient descent, among others. All those strategies minimize the errors of the graph, and should obtain the same optimal result if the SLAM problem was strictly convex. However this is not the case due to the usual high non-linearity of motion and observation models (AGARWAL, 2015).

Lu and Milios (LU; MILIOS, 1997a) are the pioneers in refining a map through the minimization of errors of a graph, however their solution includes the costly inversion of large matrices[8]. Gutmann and Konolige (GUTMANN; KONOLIGE, 1999) propose an incremental estimation algorithm that was able to reduce the impact of the matrix in-

[7]A tridiagonal matrix is a sparse matrix whose non-zero entries are confined to the main diagonal and to the first diagonals above and below the main diagonal.

[8]Increasing the number of graph nodes implies in a cubic increase in the cost of the matrix inversion required to solve the optimization problem (LU; MILIOS, 1997a).

Figure 2.9: Pose graph optimization in a dataset recorded at MIT Killian Court. Figure extracted from (GRISETTI et al., 2010b).



(a) Before optimization (b) After optimization

versions. Dellaert and Kaess (DELLAERT; KAESS, 2006) propose the method called $\sqrt{SAM}$ (square root *smoothing and mapping*), which is the first method to exploit matrix sparsity by performing factorizations, such as QR, LU and Cholesky decomposition. By avoiding ordinary matrices inversions, their method is able to solve the optimization problem much more efficiently than the previous methods in literature. Frese (FRESE, 2006) proposes Treemap, which improves sparsification by ignoring weak correlations between distant nodes. Kaess et al. propose $iSAM$ (KAESS; RANGANATHAN; DELLAERT, 2007), an incremental version of $\sqrt{SAM}$ that performs partial reorderings in the sparse matrix to improve the factorization efficiency, and $iSAM2$ (KAESS et al., 2011), an improved version of $iSAM$ using a more efficient data structure.

Another popular approach is to minimize errors through relaxation techniques such as stochastic gradient descent, as proposed by Olson et al. (OLSON; LEONARD; TELLER, 2006). The idea of relaxation methods, which are generally more robust to poor initial estimates than the least-squares approaches, is to move the nodes, one at a time, to reduce the errors. Grisetti et al. (GRISETTI et al., 2007) propose TORO, an extension of Olson's method, that uses a tree parametrization of nodes to increase efficiency. Lastly, there are hierarchical methods that improve the optimization efficiency by dividing the process into multiple levels of submaps (BOSSE et al., 2003; ESTRADA; NEIRA; TARDOS, 2005; NI; DELLAERT, 2010; GRISETTI et al., 2010a).

Due to the high complexity of the problem and the independence between front-end and back-end, some researchers have been focusing on such questions separately. In this thesis, we focus on the front-end, introducing a novel approach based on techniques from linguistic processing. As back-end, we use a traditional Gauss-Newton approach, which will be detailed next.

2.1.3.1 Pose graph optimization via Gauss-Newton

We present a traditional graph-based SLAM back-end using the Gauss-Newton algorithm and matrix factorization, as described in (GRISETTI et al., 2010b). See AppendixB for the complete derivation of the algorithm, and more implementation details. In short, the core of Gauss-Newton algorithm is to minimize the errors of the arcs, i.e., the errors between the observed and the expected relative poses between nodes.

The **observed relative pose**, $\boldsymbol{r}(\boldsymbol{a}_{ij})$, of node $\boldsymbol{x}_j$ as viewed from node $\boldsymbol{x}_i$ is obtained by the SLAM front-end (e.g. analyzing odometry readings or finding matches of observations). On the other hand, the **expected relative pose** $\hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s})$ is generated by the transformation between the current poses of nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the state $\boldsymbol{s}$,

$$\hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s}) = \boldsymbol{x}_j \ominus \boldsymbol{x}_i, \tag{2.3}$$

where the operator $\ominus$ computes the relative pose between two poses[9].

The definition of **error**, $\boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})$, associated to the arc $\boldsymbol{a}_{ij}$ connecting the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ at the current state $\boldsymbol{s}$ (as illustrated in Figure 2.10) is given by the relative pose between the observed and the expected relative poses:

$$\boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s}) = \hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s}) \ominus \boldsymbol{r}(\boldsymbol{a}_{ij}), \tag{2.4}$$

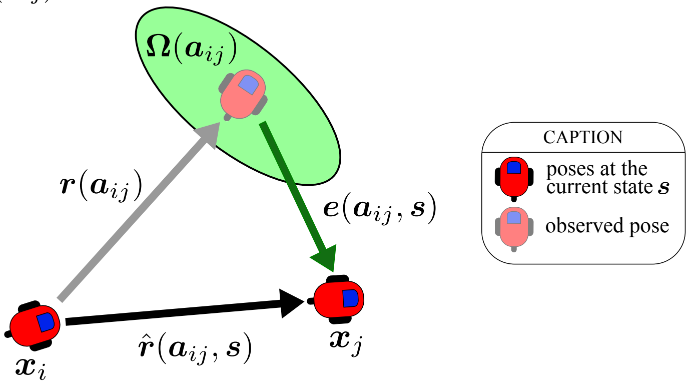
An algorithm for optimizing a pose graph must search for the configuration of robot states (nodes) that minimizes the errors of the graph arcs. Defining the sum of squared errors of all the observations as

$$F(\boldsymbol{s}) = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})^T \, \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \, \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s}), \tag{2.5}$$

[9]We define the operator $\ominus$, that computes the relative pose of $\boldsymbol{x}_b$ in relation to $\boldsymbol{x}_a$, as follows:

$$\boldsymbol{x}_b \ominus \boldsymbol{x}_a = \begin{pmatrix} \cos(\theta(\boldsymbol{x}_a)) & \sin(\theta(\boldsymbol{x}_a)) & 0 \\ -\sin(\theta(\boldsymbol{x}_a)) & \cos(\theta(\boldsymbol{x}_a)) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x(\boldsymbol{x}_b) - x(\boldsymbol{x}_a) \\ y(\boldsymbol{x}_b) - y(\boldsymbol{x}_a) \\ \theta(\boldsymbol{x}_b) - \theta(\boldsymbol{x}_a) \end{pmatrix}$$

Figure 2.10: Error between observed relative pose $\boldsymbol{r}(\boldsymbol{a}_{ij})$ of node $\boldsymbol{x}_j$ from node $\boldsymbol{x}_i$ and expected relative pose $\hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s})$. The uncertainty associated to $\boldsymbol{a}_{ij}$ is given by the information matrix $\boldsymbol{\Omega}(\boldsymbol{a}_{ij})$.


$\boldsymbol{\Omega}(\boldsymbol{a}_{ij})$
$\boldsymbol{r}(\boldsymbol{a}_{ij})$
$\boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})$
$\hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s})$
$\boldsymbol{x}_i$
$\boldsymbol{x}_j$
CAPTION
poses at the current state $\boldsymbol{s}$
observed pose

the optimal configuration $\boldsymbol{s}^*$ that leads to the minimum $F(\boldsymbol{s})$ is

$$\boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmin}}\, F(\boldsymbol{s}). \tag{2.6}$$

Equation 2.6 can be solved through Gauss-Newton algorithm by applying successive linearizations of the error around a guess $\breve{\boldsymbol{s}}$. The linearization is done through a first-order Taylor series expansion,

$$\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) \simeq \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{s} \tag{2.7}$$

where $\boldsymbol{J}_{ij}$ is the Jacobian of the error computed at $\breve{\boldsymbol{s}}$, which is non-zero only in the blocks associated to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and it has a $3 \times 3n$ dimension (i.e. $\boldsymbol{J}_{ij}$ has a 3×3 block of partial derivatives for each one of the n nodes):

$$\boldsymbol{J}_{ij} = \left.\frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{s}}\right|_{\boldsymbol{s}=\breve{\boldsymbol{s}}} = \left(\boldsymbol{0} \cdots \boldsymbol{0} \;\; \underbrace{\boldsymbol{A}_{ij}}_{\frac{\partial \boldsymbol{e}_{ij}(\boldsymbol{s})}{\partial \boldsymbol{x}_i}} \;\; \boldsymbol{0} \cdots \boldsymbol{0} \;\; \underbrace{\boldsymbol{B}_{ij}}_{\frac{\partial \boldsymbol{e}_{ij}(\boldsymbol{s})}{\partial \boldsymbol{x}_j}} \;\; \boldsymbol{0} \cdots \boldsymbol{0} \right) \tag{2.8}$$

We approximate $F(\boldsymbol{s})$ with $F(\breve{\boldsymbol{s}} + \Delta\boldsymbol{s})$ by applying the results of Equation 2.7 in Equation 2.5:

$$F(\breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \left(\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}} + \Delta\boldsymbol{s})^T \, \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \, \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) \right) \tag{2.9}$$

$$\simeq c + 2\boldsymbol{b}^T\Delta\boldsymbol{s} + \Delta\boldsymbol{s}^T\boldsymbol{H}\Delta\boldsymbol{s} \tag{2.10}$$

where

$$
\begin{aligned}
c &= \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} c_{ij} = \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) \\
\boldsymbol{b}^T &= \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \boldsymbol{b}_{ij}^T = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{J}_{ij} \\
\boldsymbol{H} &= \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \boldsymbol{H}_{ij} = \boldsymbol{J}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{J}_{ij}
\end{aligned}
$$

We find the optimal $\Delta\boldsymbol{s}^*$ that minimizes the approximated $F(\breve{\boldsymbol{s}} + \Delta\boldsymbol{s})$ by solving the linear system

$$\boldsymbol{H}\Delta\boldsymbol{s}^* = -\boldsymbol{b}. \tag{2.11}$$

Finally, we obtain the resulting state by adding $\Delta\boldsymbol{s}^*$ to the initial guess.

$$\boldsymbol{s}^* = \breve{\boldsymbol{s}} + \Delta\boldsymbol{s}^* \tag{2.12}$$

An important aspect of Equation 2.11 is that $\boldsymbol{H}$ is the information matrix of the full state, obtained using the Jacobians to project the measurement error into the trajectories space. Additionally, $\boldsymbol{H}$ is a sparse matrix, that allows the use of efficient techniques to solve the problem, such as Cholesky factorizations[10]. The sparsity of $\boldsymbol{H}$ is consequence of the product of the Jacobians, which are only non-zero in a couple of blocks, as shown in Equation 2.8.

The outline of the iterative Gauss-Newton algorithm for optimizing a pose graph is presented in Algorithm 2.3. For each constraint $\boldsymbol{a}_{ij}$ of the graph, we first compute the error $\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})$ and the Jacobians' non-zero components $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ (lines 5-7). Then, we update only the corresponding blocks[11] of matrix $\boldsymbol{H}$ and vector $\boldsymbol{b}$ (lines 8-13). Before solving the linear system, we must fix in zero the variation of one node (line 14), otherwise there would be infinite solutions to the linear system and the algorithm would not converge. To do this we add the identity matrix to the block $\boldsymbol{H}_{[00]}$ corresponding to the first node (any node can be chosen). The linear system is solved (e.g. via Cholesky factorization) and the initial guess is updated (lines 15-16). After the convergence of the solution, the first node is released (line 17) and the robot state is updated (lines 18-19).

[10]We used the efficient implementation of Cholesky factorization from the Eigen library (GUENNEBAUD; JACOB et al., 2010)

[11]We use the notation $\boldsymbol{H}_{[ij]}$ to indicate the 3×3 block from matrix $\boldsymbol{H}$ that is respectively associated to the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ (the block goes from row $3i$ to row $3i + 2$ and from column $3j$ to column $3j + 2$). Similarly, the notation $\boldsymbol{b}_{[i]}$ indicates the 3×1 block from vector $\boldsymbol{b}$ that is associated to node $\boldsymbol{x}_i$ (the block goes from row $3i$ to row $3i + 2$).

Algorithm 2.3: Gauss-Newton algorithm for pose graph optimization

```
Input: $\breve{\boldsymbol{s}}, \mathcal{A}$
Output: $\boldsymbol{s}^*$, $\boldsymbol{H}^*$
1  while $\neg$ converged do
2      $\boldsymbol{H} = 0$
3      $\boldsymbol{b} = 0$
4      for $\boldsymbol{a}_{ij}$ in $\mathcal{A}$ do
5          $\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) = \hat{\boldsymbol{r}}(\boldsymbol{a}_{ij}, \boldsymbol{s}) \ominus \boldsymbol{r}(\boldsymbol{a}_{ij})$
6          $\boldsymbol{A}_{ij} = \left.\frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{s}=\breve{\boldsymbol{s}}}$
7          $\boldsymbol{B}_{ij} = \left.\frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{x}_j}\right|_{\boldsymbol{s}=\breve{\boldsymbol{s}}}$
8          $\boldsymbol{H}_{[ii]} \mathrel{+}= \boldsymbol{A}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{A}_{ij}$
9          $\boldsymbol{H}_{[ij]} \mathrel{+}= \boldsymbol{A}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{B}_{ij}$
10         $\boldsymbol{H}_{[ji]} \mathrel{+}= \boldsymbol{B}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{A}_{ij}$
11         $\boldsymbol{H}_{[jj]} \mathrel{+}= \boldsymbol{B}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{B}_{ij}$
12         $\boldsymbol{b}_{[i]} \mathrel{+}= \boldsymbol{A}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})$
13         $\boldsymbol{b}_{[j]} \mathrel{+}= \boldsymbol{B}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})$
14     $\boldsymbol{H}_{[00]} \mathrel{+}= \boldsymbol{I}_{3\times3}$
15     $\Delta\boldsymbol{s}^* = solve(\boldsymbol{H}\Delta\boldsymbol{s}^* = -\boldsymbol{b})$
16     $\breve{\boldsymbol{s}} \mathrel{+}= \Delta\boldsymbol{s}^*$
17 $\boldsymbol{H}_{[00]} \mathrel{-}= \boldsymbol{I}_{3\times3}$
18 $\boldsymbol{s}^* = \breve{\boldsymbol{s}}$
19 $\boldsymbol{H}^* = \boldsymbol{H}$
20 return $\boldsymbol{s}^*$, $\boldsymbol{H}^*$
```

2.2 Kernel Density Estimation on Images

Density estimation is the process of estimating the probability density function f of a random variable X, given a set of observed data points sampled from f (SILVERMAN, 1986). Kernel density estimation[12] (KDE), $\hat{f}_h(x)$, is a non-parametric estimate of f computed at point x, defined as

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i), \tag{2.13}$$

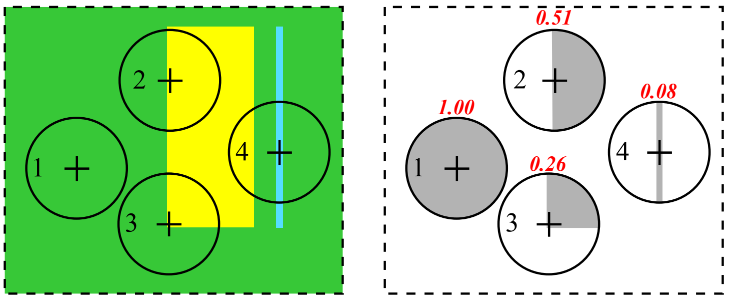
where $K_h(d)$ is a circular kernel function that operates over all points at distance $d \leq h$ from x, where h is called the bandwidth of the kernel.

KDE is widely used in applications of numerous fields such as data analysis, pattern recognition and image processing (SCOTT, 1992; SMITH; BRADY, 1997; ELGAM-

[12]Further details on kernel density estimation are presented in AppendixC.

MAL et al., 2002; MITTAL; PARAGIOS, 2004; LIU; ZHANG; YOU, 2007; JORGE, 2012). Elgammal et al. (ELGAMMAL et al., 2002) use kernel density estimation for background and foreground detection on image sequences. They estimate, for each pixel of an image, the probability of belonging to foreground/background by computing a KDE using the color intensity variation over time, that is, the KDE is composed of the same pixel at successive instants of time. High density values result from low intensity variations (i.e. a pixel with constant color), which is a property of background elements. Conversely, low density values are associated to moving objects, which are considered foreground elements. Defining a good kernel bandwidth is a problem of KDE, thus one possible approach to circumvent this issue is using variable-size kernels. Mittal and Paragios (MITTAL; PARAGIOS, 2004) propose a method that combines two simple strategies: adapting bandwidth at each estimation point x ("balloon estimator") and for each data point x_i ("sample-point estimator").

Figure 2.11: Computing kernel density estimates in four different positions of an image. In gray we illustrate the USAN areas (pixels with similar color to the center pixel of the mask).



(a) Original image (b) Density estimates

Our work is inspired on techniques that use KDE for feature detection on images, dating back to the seminal paper of Smith and Brady that introduces the SUSAN principle for edges and corners detection (SMITH; BRADY, 1997), which is illustrated in Figure 2.11. Kernel density estimates are computed applying a circular mask (circle with black border) over the pixels of an image, as shown in (a). In the example, the mask is applied at four different positions (1,2,3,4). At each position, the pixels inside the mask are compared to the "nucleus", i.e. the center pixel of the mask (black cross), in terms of a given attribute, e.g. color intensity. The kernel density estimate is defined by the ratio between the area of pixels of similar attribute, known as the "*Univalue Segment Assimilating Nucleus*" or USAN (shown in gray in Figure 2.11(b)), and the total area of the kernel. The USAN area is at maximum when the mask is over a uniform region of the image surface (position 1), it decreases to 50% near the straight edge of a wide region (position 2), to around 25%

just inside a corner (position 3), and to near 0% over thin lines (position 4). The smallest USAN area – or simply SUSAN – occurs exactly over edges or corners, therefore it is a powerful tool for feature detection.

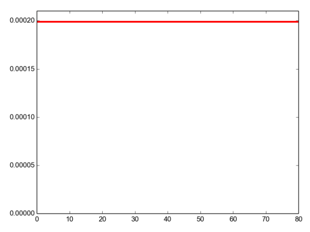
Different kernel profiles can be used depending on the application to enhance particular aspects of the estimates. Among the variety of kernel profiles present in the literature, two of the most popular are the uniform kernel, UK_h, and the Gaussian kernel, GK_h, defined as

$$UK_h(d) = \begin{cases} a & \text{, if } d \leq h \\ 0 & \text{, otherwise} \end{cases} , \tag{2.14}$$

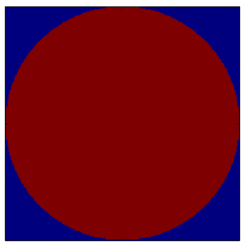
$$GK_h(d) = \frac{1}{\sqrt{2\pi h^2}} e^{-\frac{1}{2}\frac{d^2}{h^2}}, \tag{2.15}$$

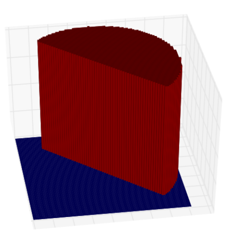
where d is the distance $||x - x_i||$ from a cell x_i to the kernel nucleus x, and a is the height of the uniform kernel – typically $a = \frac{1}{2h}$ in 1D and $a = \frac{1}{\pi h^2}$ in 2D. The uniform kernel gives the same importance to all cells inside the kernel mask that share the same attribute with the nucleus. Therefore the result of a KDE using the uniform kernel is basically the count of similar cells, which corresponds to an approximation of the area occupied by those cells. On the other hand, the Gaussian kernel gives more importance to cells near the center of the kernel, which reduces noise influence. Figures 2.12 and 2.13 show the uniform and the Gaussian kernel, respectively, where (a) in each figure is the one-dimensional kernel profile, (b) is the two-dimensional kernel profile, and (c) is a $2\,{}^1\!/_2$ D representation of the two-dimensional kernel profile.

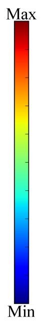
Figure 2.12: Uniform kernel.



(a) 1D kernel

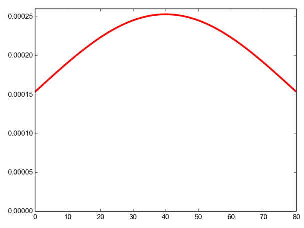

(b) 2D kernel


(c) Half view in $2\,{}^1\!/_2$ D

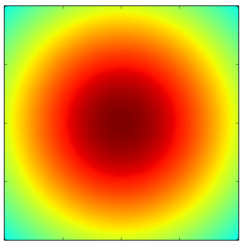


Liu and Zhang apply Gaussian kernel density estimation to detect wide lines in many different applications, such as detection of palm print lines, tongue cracks and vascular stenosis in x-ray images (LIU; ZHANG, 2005; LIU; ZHANG; YOU, 2007; LIU; ZHANG, 2008). Beyond structural information given by traditional edges detection techniques, the aforementioned applications require width information, that can be estimated by the density of pixels with similar attribute on a line.

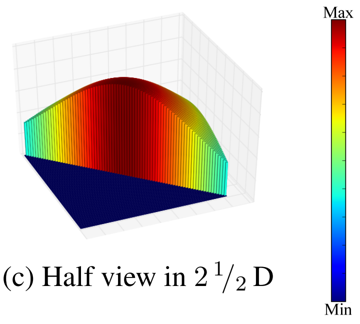
Figure 2.13: Gaussian kernel.



(a) 1D kernel



(b) 2D kernel



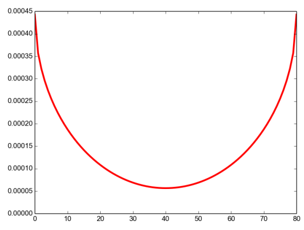
(c) Half view in $2\,{}^{1}\!/_{2}$ D

Jorge (JORGE, 2012) proposes a wide line detector technique using an inverted kernel profile, IK_h, which gives more weight to cells in the border of the kernel. The kernel is defined by the subtraction of an oblate ellipsoid from the uniform circular kernel,

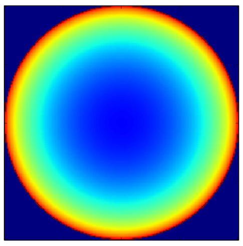
$$IK_h(d) = \begin{cases} h(b+c) - \left(\sqrt{h^2 - d^2}\,\right)c & \text{, if } d \leq h \\ 0 & \text{, otherwise} \end{cases} \tag{2.16}$$

where $b+c$ is the height of the uniform circular kernel and c is the length of the semi-axis of the oblate ellipsoid, with $b \approx 0.871c$. Figure 2.14 illustrates the inverted kernel profile in 1D and 2D.

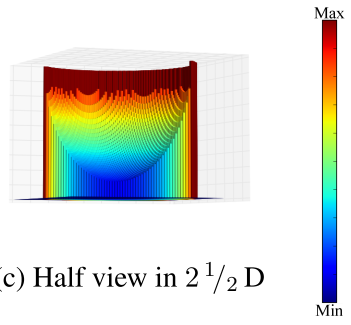
Figure 2.14: Inverted kernel.



(a) 1D kernel



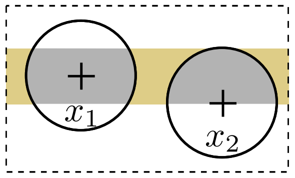
(b) 2D kernel



(c) Half view in $2\,{}^{1}\!/_{2}$ D

To illustrate the motivation behind IK_h, consider a kernel with bandwidth h and two points in line, x_1 and x_2, as depicted in Figure 2.15. The point x_1 is at the center of the line and the point x_2 is at its border. The densities computed using UK_h or GK_h with bandwidth h will result in different values for both points, i.e. $\hat{f}_h(x_1) > \hat{f}_h(x_2)$, as shown in the table in Figure 2.15(b). This is true for any monotonic decreasing kernel. However, this may not be true for monotonic increasing kernels, such as IK_h. Thus, Jorge defines the values of b and c in Equation 2.16 such that $\hat{f}_h(x_1) = \hat{f}_h(x_2)$, that is, the densities computed as the kernel moves towards the center of the line tend to stay constant.

Figure 2.15: Kernel density estimation made with IK_h (a monotonic decreasing kernel profile) obtains the same result in points x_1 and x_2, while monotonic increasing kernel profiles $\left(UK_h \text{ and } GK_h\right)$ obtain different results. (a) Kernels at two points in a wide line. (b) Estimates obtained with UK_h, GK_h and IK_h.



(a)

	UK_h	GK_h	IK_h
$\hat{f}_h(x_1)$	0.61	0.64	**0.50**
$\hat{f}_h(x_2)$	0.50	0.50	**0.50**

(b)

2.3 n-grams: an efficient technique from shallow linguistic processing

Speech and language processing is the study of techniques that process human language, encompassing applications such as word counting, text parsing, spoken and writen language translation, etc (MANNING; SCHÜTZE, 1999). The knowledge of language used as the basis for such applications can be separated into six categories: *Phonetics and Phonology*, i.e. the study of linguistic sounds; *Morphology*, i.e. the study of the meaningful components of words; *Syntax*, i.e. the study of the structural relationships between words; *Semantics*, i.e. the study of meaning; *Pragmatics*, i.e. the study of how language is used to accomplish goals; and *Discourse*, i.e. the study of linguistic units larger than a single utterance (JURAFSKY; MARTIN, 2000).

Solving ambiguities at one of those six levels is the basic problem of most tasks in this area. Jurafsky and Martin illustrate the different types of ambiguities in human language by using the following simple sentence: *"I made her duck"* (JURAFSKY; MARTIN, 2000). There are several possible interpretations for this sentence: *I cooked a duck (aquatic bird) for her; I cooked a duck belonging to her; I somehow created the duck belonging to her; I caused her to quickly get down; or I magically turned her into a bird.* As we can see, the word *duck* can be a verb or a noun, the word *her* can be a dative or a possessive pronoun, the word *made* is a verb that can take a single direct object or take two objects or take another verb. Ambiguities like these happen because analyzing only a few words out of context is usually not enough.

For decades, these problems have been studied in several science fields such as linguistics, computer science, electrical engineering and psychology (MANNING; SCHÜTZE, 1999; JURAFSKY; MARTIN, 2000). The origins of those studies date back to the half of the 20th century with the seminal works on automata, regular expressions (SHANNON, 1948; KLEENE, 1956), and formal language theory (CHOMSKY, 1956; BACKUS, 1959;

NAUR et al., 1963). Shannon, which applied probabilistic models of discrete Markov process to automata for language processing, brought the concept of entropy from thermodynamics to measure the information content of a language. He measured the entropy of English[13] using n-grams, which were defined by Shannon as sequences of n letters (SHANNON, 1951).

An n-gram is a contiguous sequence of n elements from a given text, where the elements can be words, syllables, letters (as originally defined by Shannon), etc. Using n-grams is one of the most straightforward techniques in language processing: it does not perform any analysis of morphology, syntax or semantics. On the contrary, it is only based on counting occurrences of sequences of elements in a *corpus*[14] (JURAFSKY; MARTIN, 2000).

Table 2.1 shows examples of n-grams extracted from the sentence: *"the ghost of electricity howls in the bones of her face."*[15]. In the top table, a gram corresponds to a letter, thus a unigram (1-gram) is just one letter, bigrams (2-grams) are sequences of two letters, trigrams (3-grams) are sequences of three letters, etc. In the bottom table, a gram correspond to a word, that is, in this case n-grams are sequences of n words. When using letter-based n-grams, the total number of grams (TG) extracted from the text is larger than when using word-based n-grams. However, the amount of possible grams is much smaller, thus the total number of repeated grams (TRG) is substantially larger. The same situation happens when using large sequences of grams (independently from the gram type). For instance, the sequence of letters '*t-h-e*' is usually much less common than the single letter '*t*' (in fact, it is at least as common). Likewise, the single word "*of*" is obviously more common, in general, than the sequence "*of electricity howls*". In this thesis proposal a gram correspond to a word.

n-Grams are widely used for word identification and prediction, which are fundamental for tasks such as speech recognition and spelling error detection. Naive spell checkers, which just verify the correctness of words by searching for them in a dictionary, do not always work properly because many times spelling errors result in real words due to ambiguities in the language. In such cases, the context of the words, that can be obtained from the previous sequence of words, gives us an indication of the word correctness.

The basic idea behind the use of n-grams is to assign probabilities to every sequence of

[13]Shannon experiments were made over the 484-pages book "Jefferson the Virginian" (1948), part of the series "Jefferson and his time" by Dumas Malone, and the word frequency tables in (DEWEY, 1923).

[14]*corpus* is a collection of texts – plural: *corpora*.

[15]Excerpt from the song "*Visions of Johanna*" from the album *Blonde on Blonde* (1966) by Bob Dylan.

Table 2.1: Example of n-grams extracted from a sentence, with n varying from 1 to 3 (i.e. unigrams, bigrams and trigrams). Grams are defined by letters in the top table, and by words in the bottom table. For greater ease of understanding, each different gram is associated to a unique (superscripted) index. First occurrences of grams are shown in black, while repetitions are shown in red. The total number of grams (TG) and the total number of repeated grams (TRG) are presented for each configuration.

Sentence

"the ghost of electricity howls in the bones of her face."

Letter n-Grams

n	Grams	TG	TRG
1	t $^{(1)}$, h $^{(2)}$, e $^{(3)}$, _ $^{(4)}$, g $^{(5)}$, **h** $^{(2)}$, o $^{(6)}$, s $^{(7)}$, **t** $^{(1)}$, **_** $^{(4)}$, **o** $^{(6)}$, f $^{(8)}$, **_** $^{(4)}$, **e** $^{(3)}$, l $^{(9)}$, **e** $^{(3)}$, c $^{(10)}$, **t** $^{(1)}$, r $^{(11)}$, i $^{(12)}$, **c** $^{(10)}$, **i** $^{(12)}$, **t** $^{(1)}$, y $^{(13)}$, **_** $^{(4)}$, **h** $^{(2)}$, **o** $^{(6)}$, w $^{(14)}$, **l** $^{(9)}$, **s** $^{(7)}$, **_** $^{(4)}$, **i** $^{(12)}$, n $^{(15)}$, **_** $^{(4)}$, **t** $^{(1)}$, **h** $^{(2)}$, **e** $^{(3)}$, **_** $^{(4)}$, b $^{(16)}$, **o** $^{(6)}$, **n** $^{(15)}$, **e** $^{(3)}$, **s** $^{(7)}$, **_** $^{(4)}$, **o** $^{(6)}$, **f** $^{(8)}$, **_** $^{(4)}$, **h** $^{(2)}$, **e** $^{(3)}$, **r** $^{(11)}$, **_** $^{(4)}$, **f** $^{(8)}$, a $^{(17)}$, **c** $^{(10)}$, **e** $^{(3)}$	55	38
2	th $^{(1)}$, he $^{(2)}$, e_ $^{(3)}$, _g $^{(4)}$, gh $^{(5)}$, ho $^{(6)}$, os $^{(7)}$, st $^{(8)}$, t_ $^{(9)}$, _o $^{(10)}$, of $^{(11)}$, f_ $^{(12)}$, _e $^{(13)}$, el $^{(14)}$, le $^{(15)}$, ec $^{(16)}$, ct $^{(17)}$, tr $^{(18)}$, ri $^{(19)}$, ic $^{(20)}$, ci $^{(21)}$, it $^{(22)}$, ty $^{(23)}$, y_ $^{(24)}$, _h $^{(25)}$, **ho** $^{(6)}$, ow $^{(26)}$, wl $^{(27)}$, ls $^{(28)}$, s_ $^{(29)}$, _i $^{(30)}$, in $^{(31)}$, n_ $^{(32)}$, _t $^{(33)}$, **th** $^{(1)}$, **he** $^{(2)}$, **e_** $^{(3)}$, _b $^{(34)}$, bo $^{(35)}$, on $^{(36)}$, ne $^{(37)}$, es $^{(38)}$, **s_** $^{(29)}$, **_o** $^{(10)}$, **of** $^{(11)}$, **f_** $^{(12)}$, **_h** $^{(25)}$, **he** $^{(2)}$, er $^{(39)}$, r_ $^{(40)}$, _f $^{(41)}$, fa $^{(42)}$, ac $^{(43)}$, ce $^{(44)}$	54	10
3	the $^{(1)}$, he_ $^{(2)}$, e_g $^{(3)}$, _gh $^{(4)}$, gho $^{(5)}$, hos $^{(6)}$, ost $^{(7)}$, st_ $^{(8)}$, t_o $^{(9)}$, _of $^{(10)}$, of_ $^{(11)}$, f_e $^{(12)}$, _el $^{(13)}$, ele $^{(14)}$, lec $^{(15)}$, ect $^{(16)}$, ctr $^{(17)}$, tri $^{(18)}$, ric $^{(19)}$, ici $^{(20)}$, cit $^{(21)}$, ity $^{(22)}$, ty_ $^{(23)}$, y_h $^{(24)}$, _ho $^{(25)}$, how $^{(26)}$, owl $^{(27)}$, wls $^{(28)}$, ls_ $^{(29)}$, s_i $^{(30)}$, _in $^{(31)}$, in_ $^{(32)}$, n_t $^{(33)}$, _th $^{(34)}$, **the** $^{(1)}$, **he_** $^{(2)}$, e_b $^{(35)}$, _bo $^{(36)}$, bon $^{(37)}$, one $^{(38)}$, nes $^{(39)}$, es_ $^{(40)}$, s_o $^{(41)}$, **_of** $^{(10)}$, **of_** $^{(11)}$, f_h $^{(42)}$, _he $^{(43)}$, her $^{(44)}$, er_ $^{(45)}$, r_f $^{(46)}$, _fa $^{(47)}$, fac $^{(48)}$, ace $^{(49)}$	53	4

Word n-Grams

n	Grams	TG	TRG
1	*the* $^{(1)}$, *ghost* $^{(2)}$, *of* $^{(3)}$, *electricity* $^{(4)}$, *howls* $^{(5)}$, *in* $^{(6)}$, ***the*** $^{(1)}$, *bones* $^{(7)}$, ***of*** $^{(3)}$, *her* $^{(8)}$, *face* $^{(9)}$	11	2
2	*the ghost* $^{(1)}$, *ghost of* $^{(2)}$, *of electricity* $^{(3)}$, *electricity howls* $^{(4)}$, *howls in* $^{(5)}$, *in the* $^{(6)}$, *the bones* $^{(7)}$, *bones of* $^{(8)}$, *of her* $^{(9)}$, *her face* $^{(10)}$	10	0
3	*the ghost of* $^{(1)}$, *ghost of electricity* $^{(2)}$, *of electricity howls* $^{(3)}$, *electricity howls in* $^{(4)}$, *howls in the* $^{(5)}$, *in the bones* $^{(6)}$, *the bones of* $^{(7)}$, *bones of her* $^{(8)}$, *of her face* $^{(9)}$	9	0

words based on their occurrences in a *corpus*. In the statistical n-gram model, high probabilities are associated to very common sequences, while low probabilities are associated to rare sequences, which can be a good indication on possible input errors. The n-gram model can be used to predict the next word in an incomplete sentence, i.e. finding the likelihood of a word given a sequence of words. The probability of a complete sequence of t words, when considering the words occurrences in their locations as independent events, is given by $P(\mathtt{W}_1, \mathtt{W}_2, \cdots, \mathtt{W}_{t-1}, \mathtt{W}_t)$ (or simply $P(\mathtt{W}_{1:t})$), where the t-th word is represented by $\mathtt{W}_t$. This probability can be decomposed, using the chain rule, into

$$P(\mathtt{W}_{1:t}) = P(\mathtt{W}_1)P(\mathtt{W}_2|\mathtt{W}_1)P(\mathtt{W}_3|\mathtt{W}_{1:2})\cdots P(\mathtt{W}_t|\mathtt{W}_{1:t-1}). \tag{2.17}$$

The Markov assumption says that the current state of a system can be modeled based only on the previous one, which means that events too far into the past do not impact on the current state of the system. This is the definition of a first-order Markov model, however there are Markov models of higher order, i.e, models that depend not only on the previous state but on a few previous states. The n-gram model is a $(n-1)$-th order Markov model because it looks $n-1$ words into the past to estimate the probability of the current word (JURAFSKY; MARTIN, 2000).

However, computing $P(\mathtt{W}_t|\mathtt{W}_{1:t-1})$ is not feasible for large t's, because it requires counting occurrences of very long sequences, which results in extremely low probabilities due to the huge number of word combinations. Hence, n-grams adopt the Markov assumption to approximate the probability of a word given all previous words to the probability given only the n previous words,

$$P(\mathtt{W}_t|\mathtt{W}_{1:t-1}) \approx P(\mathtt{W}_t|\mathtt{W}_{t-n:t-1}). \tag{2.18}$$

Besides many applications of n-grams in text prediction (SHANNON, 1951; REITHINGER et al., 1996; GARAY-VITORIA; ABASCAL, 2006), they also have been used in compression algorithms (MILLAR et al., 2006; PAULS; KLEIN, 2011), information retrieval (WANG; MCCALLUM; WEI, 2007; SIDOROV et al., 2014), activities prediction (ARNDT et al., 2013), plagiarism detection (BARRÓN-CEDEÑO; ROSSO, 2009; STAMATATOS, 2011), among other topics. In our work, n-grams are used for approximate matching of word sequences representing the environment observed by a robot. In other words, we use n-grams for detecting loops in the simultaneous localization and mapping problem.

3 FREE-SPACE DENSITY (FSD): TRANSLATING RAW SENSOR MEASUREMENTS INTO SIMPLE OBSERVATION VALUES

3.1 Introduction

Choosing an adequate world representation is an important aspect that should be considered in tasks of robotics. For instance, we can describe an environment using a set of points extracted from laser readings, or with an arrangement of features extracted from camera images, however, such approaches can be too costly and unsuitable for dealing with highly ambiguous scenarios. Sometimes it is interesting to describe an environment using more abstract information, like '*the robot is in a long and narrow region*' or '*the robot is entering a wide space*', etc. For obtaining this type of higher-level knowledge about the environment, we need a compact and efficient low-level representation of sensors information.

In this chapter[1], we propose an observation model that estimates, what we denominated, the Free-Space Density (FSD) of a region. By applying a kernel density estimate, as described in Section 2.2, over the local map surrounding the robot, we obtain a single-valued density that represents the amount of free-space in that location – where low densities are associated to tight/narrow spaces, and high densities are associated to large empty spaces. We evaluate our proposal in a global localization case study using particle filters. Given that the FSD measure is independent of orientation, it is possible to implement an efficient pre-caching step in order to improve the performance of the process. We show through experiments in comparison with traditional approaches that our method is efficient and effective in the tested scenarios.

3.2 The Free-Space Density

Our novel observation model estimates the Free-Space Density (FSD) of a given region surrounding a cell m_0 of the environment by computing a kernel density estimate, Ψ,

[1] The core idea of this chapter was originally published in the paper "Fast Monte Carlo Localization using spatial density information" (MAFFEI et al., 2015a), presented at the 2015 IEEE International Conference on Robotics and Automation (ICRA'15).

centered on $\boldsymbol{m}_0$ by

$$\Psi(\boldsymbol{m}_0) = \sum_{\boldsymbol{m}_i} s(\boldsymbol{m}_i, \boldsymbol{m}_0) K(||\boldsymbol{m}_i - \boldsymbol{m}_0||) \tag{3.1}$$

where

$$s(\boldsymbol{m}_i, \boldsymbol{m}_0) = \begin{cases} 1 & \text{, if } \boldsymbol{m}_i \text{ is a cell that belongs to the free space region connected to } \boldsymbol{m}_0 \\ 0 & \text{, otherwise} \end{cases} \tag{3.2}$$

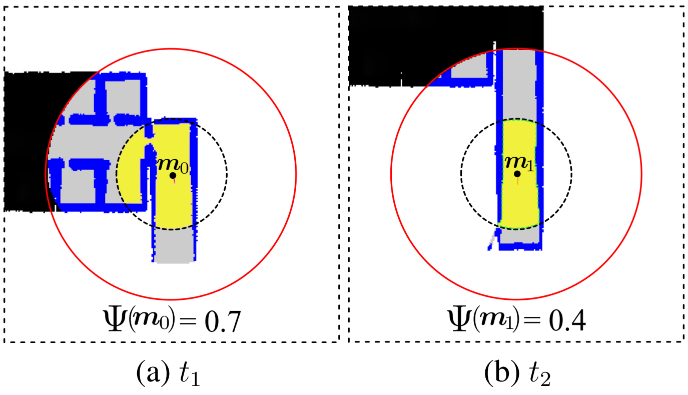
The first step required for computing the FSD in the robot location is to build a local grid map of the robot surroundings. Then, we apply a simple flood-fill algorithm to determine all cells, inside a given kernel radius, that are connected to the cell at the robot position. A kernel density estimate is computed using these cells, and depending on the chosen kernel profile, different results can be obtained. We will discuss these topics in details next.

3.2.1 Building a local map

Inaccuracies in the mapping process arise from errors in the robot motion and sensor observations. Usually the observation model for range finder sensors such as laser is highly precise, but the same is generally not true for the motion model of mobile robots. In fact, we can say that a local map obtained by a discrete set of laser measurements and few odometry readings is locally consistent. This assumption is supported by evidences found in previous works of simultaneous localization and mapping (BOSSE et al., 2003; MAFFEI et al., 2013).

We build the local map using the simple and efficient Histogramic In-Motion Mapping technique (HIMM) (BORENSTEIN; KOREN, 1991), detailed in Section 2.1.2, but any standard grid mapping technique can be used (MORAVEC; ELFES, 1985; MURPHY, 2000; HAHNEL et al., 2003b). Our approach only keeps cell information up to a certain maximum radius, r_{max}. Once the robot moves (along with the center of the local map), we disregard the information of distant cells, i.e. cells that are positioned farther than r_{max} from the robot, to keep map consistency (see Figure 3.1).

Figure 3.1: Example of FSD values, in two different time steps. The red circle delimits the frontier of the local grid map ($r_{max} = 9m$), the black circle delimits the region under the kernel influence ($h = 4m$) and the yellow region corresponds to the visited space covered by the kernel.


m_0
$\Psi(\boldsymbol{m}_0) = 0.7$
m_1
$\Psi(\boldsymbol{m}_1) = 0.4$

(a) t_1 (b) t_2

3.2.2 Computing the free-space density

With a local map available, we can determine the free-space density using a simple flood-fill algorithm, as described in Algorithm 3.1. The density associated to a cell $\boldsymbol{m}_0$ is obtained by applying the kernel function K_h to each free-space cell, $\boldsymbol{m}_i$, within the kernel radius of bandwidth h. Free-space cells have occupancy value[2] smaller than the threshold ϵ_f, while occupied cells have occupancy value larger than the threshold ϵ_o (empirically, we set $\epsilon_f = 3$ and $\epsilon_o = 12$). The algorithm starts by adding cell $\boldsymbol{m}_0$ to a queue Q (line 1). Then it processes all cells in Q (lines 3-4) that are inside the bandwidth h of kernel K_h (lines 5-6). Each cell has its importance given by its distance d_i to the center of the kernel (line 8). The flood-fill algorithm, computed only over free-space cells (line 7), is performed via Von Neumann neighborhood (i.e. four-neighborhood: $\uparrow$, $\rightarrow$, $\downarrow$, $\leftarrow$) (line 9-10). An important issue occurs when the expansion reaches unknown cells, that is, when $\epsilon_f \leq occ_{\boldsymbol{m}_i} \leq \epsilon_o$ (lines 11-12). In this case, we know that the local map is incomplete, which can often happen given that we use a range finder with a $180°$ field-of-view, and the density estimate is possibly smaller than the one obtained with a fully known map. For this situation we designed two different free-space density measures that are computed for each cell:

- Ψ: the ***soft FSD***, which represents the density estimate in either complete or incomplete local maps.

[2]In the HIMM method (BORENSTEIN; KOREN, 1991), the occupancy value of each cell is an integer that varies from 0 (highest certainty of being free-space) to 16 (highest certainty of being obstacle). For more details, see Section 2.1.2.

- $\Psi^\diamond$: the ***hard FSD***, which is only defined in complete local maps. In incomplete local maps, it is marked as undefined ($\Psi^\diamond = UND_\Psi$).

When there are no reachable unknown cells inside the kernel radius, the soft FSD is exactly the same as the hard FSD (lines 13-14).

Algorithm 3.1: Free-Space Density

```
   Input: $\boldsymbol{m}_0, \boldsymbol{m}, K_h, \epsilon_f, \epsilon_o$
   Output: $\Psi$,$\Psi^\diamond$
1  $Q = [\boldsymbol{m}_0]$
2  $\Psi = \Psi^\diamond = 0$
3  while $Q \neq [\ ]$ do
4      $\boldsymbol{m}_i = pop\_front(Q)$
5      $d_i = \sqrt{(x(\boldsymbol{m}_i) - x(\boldsymbol{m}_0))^2 + (y(\boldsymbol{m}_i) - y(\boldsymbol{m}_0))^2}$
6      if $d_i < h$ then
7          if $occ_{\boldsymbol{m}_i} < \epsilon_f$ then
8              $\Psi = \Psi + K_h(d_i)$
9              foreach (non processed) neighbor $\boldsymbol{m}_k$ of $\boldsymbol{m}_i$ do
10                 $push\_back(\boldsymbol{m}_k, Q)$
11         else if $\epsilon_f \leq occ(\boldsymbol{m}_i) \leq \epsilon_o$ then
12             $\Psi^\diamond = UND_\Psi$
13 if $\Psi^\diamond \neq UND_\Psi$ then
14     $\Psi^\diamond = \Psi$
15 return $\Psi$,$\Psi^\diamond$
```

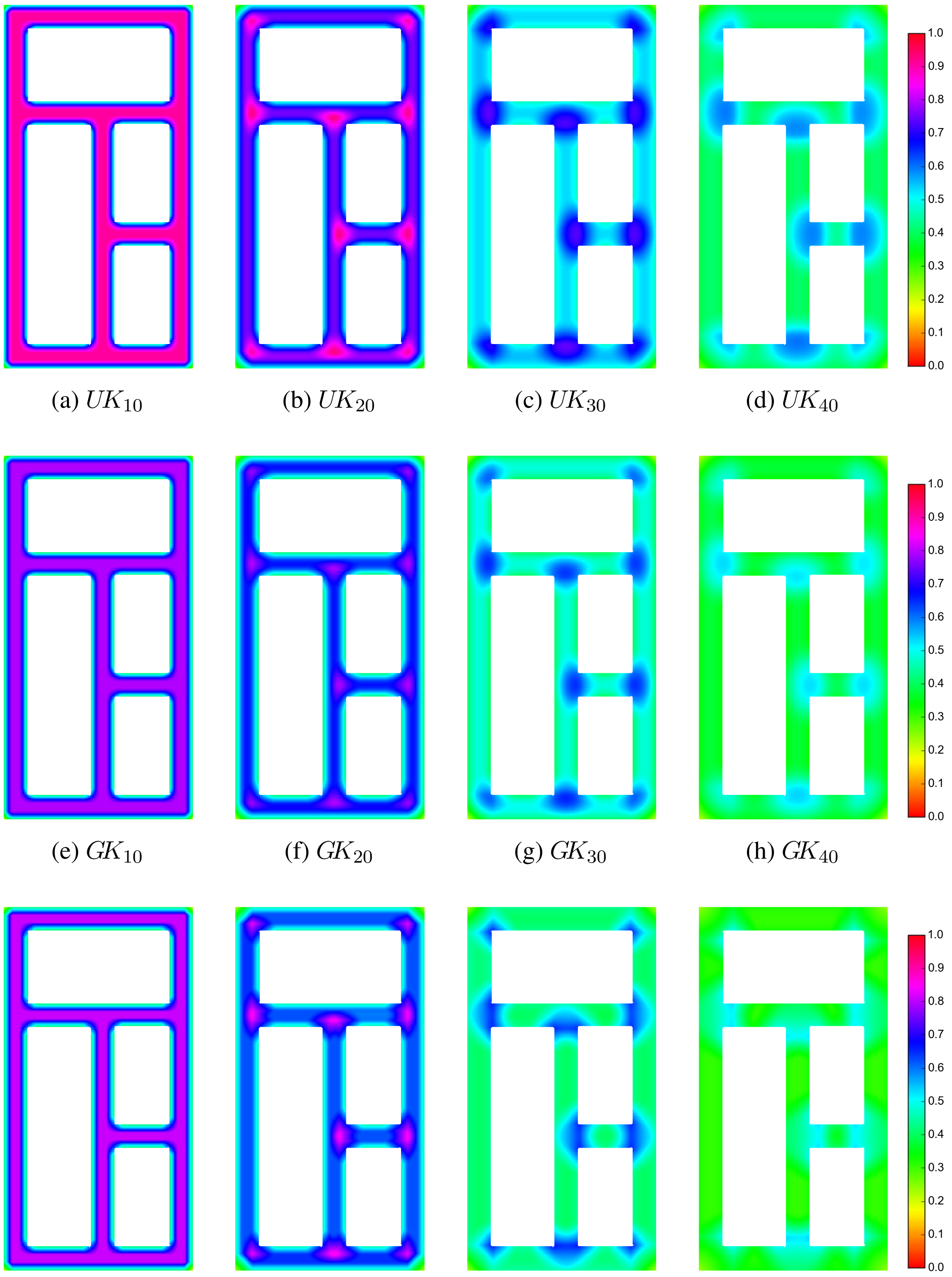
3.2.3 Using different kernel profiles to compute FSD

Different kernel profiles and kernel bandwidths can be used to compute different distributions of free-space density. Figure 3.2 shows density maps of a same environment that were built using three kernel profiles – uniform, Gaussian and inverted – and four bandwidths – 10, 20, 30 and 40 cells. Each corridor of the environment is 3 meters wide, or 30 cells, given that the grid discretization of the environment is 10 cells per meter.

We define the measure ρ, as the ratio between the kernel bandwidth (i.e. $10, 20, 30$, or 40) and the width of the corridor (i.e. 30), to classify the size of the kernel considering the environment, and examine the implications of varying the kernel bandwidth:

- *very small* ($\rho < 0.5$): exemplified by $h = 10$ cells ($\rho = 0.333$), see Figures 3.2(a)(e)(i). It is good to determine proximity to obstacles, because, most of the time, small kernels will be fully contained inside the free-space, thus the density only significantly changes close to walls.

Figure 3.2: Maps of densities varying kernel profile and kernel bandwidth. *Rows (from top to bottom):* uniform kernel, Gaussian kernel and inverted kernel. *Columns (from left to right):* bandwidths of 10, 20, 30 and 40 cells (1, 2, 3 and 4 meters).



(a) UK_{10} (b) UK_{20} (c) UK_{30} (d) UK_{40}

(e) GK_{10} (f) GK_{20} (g) GK_{30} (h) GK_{40}

(i) IK_{10} (j) IK_{20} (k) IK_{30} (l) IK_{40}

- *small* ($\rho < 1.0$): exemplified by $h = 20$ cells ($\rho = 0.666$), see Figures 3.2(b)(f)(j). It is able to identify proximity to obstacles and differentiate regions of the environment. However, it is not recommended because it often may lead to confusion, for instance, the density in the center of a corridor can be the same as the density near walls in a corner.

- *proper* ($\rho = 1.0$): exemplified by $h = 30$ cells, see Figures 3.2(c)(g)(k). It is the better choice if we want to segment the environment into different contiguous regions, such as corridors, rooms, corners and bifurcations.
- *large* ($\rho > 1.0$): exemplified by $h = 40$ cells ($\rho = 1.333$), see Figures 3.2(d)(h)(l). It is not recommended because the densities obtained with such kernels tend to be hard to differentiate. In fact, as we increase the kernel bandwidth, the FSD values in this environment decrease as a whole because the area of the kernel occupied by free-space cells becomes smaller, and therefore, it is more difficult to differentiate the places.

Regarding the kernel profile, we can say that the impact of choosing different profiles is much smaller than the impact of choosing different bandwidths. All three profiles – uniform, Gaussian and Inverted – have similar distributions considering the same bandwidth, but slightly shifted. The largest density values are usually obtained with the uniform kernel, while the smallest values are obtained with the inverted kernel. An important feature of the inverted kernel is that if we choose a bandwidth proper to the width of corridors, like $h = 30$ in this case, the obtained density is constant at every position throughout the corridor, as shown in Figure 3.2(k).

3.3 Case Study: Mobile robot localization

In this section we detail the application of the free-space density to the global localization problem of a mobile robot. Mobile robot localization is one major area of study in robotics (MAKARENKO et al., 2002). A robot cannot properly interact with the environment or execute relevant tasks without knowing its pose in the world. While simultaneous localization and mapping (SLAM) and integrated exploration techniques are able to build a map and, at the same time, to localize the robot inside this map (BOSSE et al., 2003; MAFFEI et al., 2013), such methods are sometimes unnecessary. This is the case when we already have a precise map of an environment, such as the floor plan of a building. Robots can access adapted versions of such maps prepared for mobile robot localization.

Most localization approaches model the robot pose as a probability distribution, such as Gaussians (LEONARD; DURRANT-WHYTE, 1991), histograms (BURGARD et al., 1998a) or particle filters (DELLAERT et al., 1999). In these cases, the Bayesian approach is adopted, where the previous robot state is assumed to be sufficient to infer the next one. At each step, a prediction about the robot state is made respecting the robot motion model. Then, readings from sensors are used to correct the estimate using a sensor measurement model.

The modeling using unimodal Gaussian strategies, either to track single (LEONARD; DURRANT-WHYTE, 1991) or multiple hypotheses (COX; LEONARD, 1994), usually works well with the assumption of small uncertainty. However they require mechanisms to extract salient features in the environment, such as geometric beacons (LEONARD; DURRANT-WHYTE, 1991; COX; LEONARD, 1994). In contrast, non parametric approaches, such as histogram and particle filters, can work direct with raw sensor readings. Histogram filters precompute the distance from walls in every cell of a grid representing the map, from a discrete set of orientations. They can obtain high accuracy in the localization depending on the coarseness of the grid, nonetheless they may require large amounts of memory to store all available robot states in the map, which may as well imply in prohibitive computation time. Particle filters, as defined in the Monte Carlo Localization (DELLAERT et al., 1999), solve the localization problem spreading particles (samples) throughout the map. Its cost is linear in the number of particles, but it can become expensive as the number of samples increases – e.g., in large maps. Still, particle filters are highly popular to solve the localization problem due to their underlying simplicity and robustness (THRUN; BURGARD; FOX, 2005).

The efficiency of such localization strategies relies on the way sensor observations are modeled. In turn, the measurement model of a single reading of a range finder involves a probability distribution – or a combination of probability distributions – considering parameters such as the measured distance as well as sensor position and orientation. When using a sensor that performs multiple readings at a single time step – e.g. a laser range finder – the probabilities of all readings are combined to provide a single probability of a given position in the map. So, the observation probability must be computed for each hypothesis of the robot pose. Thus, it may become costly when the number of particles is large. Some alternatives to speed-up the process are the subsampling of the measurements and the pre-computation of the ray-casting in each discrete combination of position/orientation available in the map. However, depending on the scale of the map being processed, these modifications imply in the reduction of precision and a substantial increase in memory requirements (THRUN; BURGARD; FOX, 2005; BURGARD et al., 1998a).

Our proposal basically computes and compares the free-space density in the surroundings of the robot with those in the surroundings of samples in the map. The FSD of each position in free space is precomputed and used to match the observations made by the robot in a given instant in time. Furthermore, it is orientation independent and has a low cost both in terms of processing time and memory requirements. Our method is particularly good to reach the pose tracking stage, due to the large number of samples that it can efficiently handle.

3.3.1 FSD-based Monte Carlo localization

The Monte Carlo Localization (MCL) algorithm is a simple and efficient particle filter strategy for robot localization, as described in Algorithm 2.1 (back on Section 2.1.1). It has three steps: the *sampling* of particles (pose hypotheses) based on the motion model, the *importance weighting* of each particle based on the observation model, and the *resampling* of particles based on the computed weights to approximate the posterior distribution about the robot state to the target distribution. To use our novel observation model we just have to modify the importance weighting step, that is, to define how to update the particles' weights.

In our approach, we consider the map as a discrete regular grid. Each cell in the grid map contains a free-space density value which is precomputed for future use. In other words, we apply the FSD over the entire map to obtain a scalar field of density estimates. Given that the map is fully known, the computed densities are the hard FSD ($\Psi^{\diamond}$), thus we must also compute the hard FSD over the local map built using the robot observations.

The weight w of each particle $\boldsymbol{p}_t^{[i]} = \langle \boldsymbol{x}, w \rangle$ is computed using two factors: one based on the density estimates, $f_{\Psi}(\boldsymbol{p}_t^{[i]})$, and other based on orientation estimates, $f_{\alpha}(\boldsymbol{p}_t^{[i]})$.

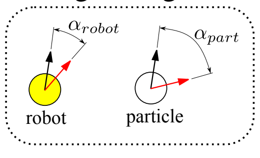
$$w(\boldsymbol{p}_t^{[i]}) = f_{\Psi}(\boldsymbol{p}_t^{[i]}) \cdot f_{\alpha}(\boldsymbol{p}_t^{[i]}) \tag{3.3}$$

The first term of the expression is computed by comparing the density value at the current robot position extracted from the local map, $\Psi^{\diamond}(\boldsymbol{m}_{robot})$, to the one associated to the current particle in the global map, $\Psi^{\diamond}(\boldsymbol{m}_{part})$,

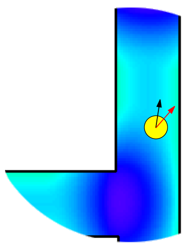
$$f_{\Psi}(\boldsymbol{p}_t^{[i]}) = \begin{cases} 1 & \text{, if } (|\Psi^{\diamond}(\boldsymbol{m}_{robot}) - \Psi^{\diamond}(\boldsymbol{m}_{part})| \leq \epsilon_d) \vee (\Psi^{\diamond}(\boldsymbol{m}_{robot}) = UND_{\Psi}) \\ 0.5 & \text{, otherwise} \end{cases} , \tag{3.4}$$

where $\boldsymbol{m}_{part} = pose2grid(\boldsymbol{x}(\boldsymbol{p}_t^{[i]}))$ is the grid cell corresponding to the particle's pose. $f_{\Psi}(\boldsymbol{p}_t^{[i]})$ is used to reduce the importance of particles with FSD values that are highly different than the FSD value observed by the robot. More specifically, we compute the difference between the FSD value of a particle, $\Psi^{\diamond}(\boldsymbol{m}_{part})$, and the FSD value measured by the robot, $\Psi^{\diamond}(\boldsymbol{m}_{robot})$, and check if this difference is larger than a density threshold ϵ_d. We set the weight of particles which failed the test as half of those which passed. Additionally, we keep the particles unaltered when the robot obtains an undefined density value (UND_{Ψ}). This situation happens just before the robot go through an opening in the environment (e.g. door), because there will be reachable unknown cells inside the kernel, which means that the measure made by the robot is not fully reliable. After the robot passes through such opening and updates those unknown cells, then the newly computed FSD will represent a proper value.

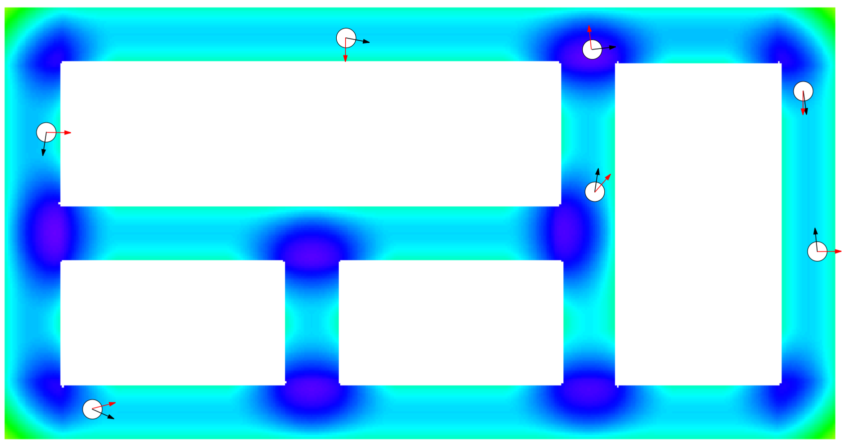
Figure 3.3: Evaluating the orientation of particles using the gradient of the FSD scalar field. (a) α_{robot} and α_{part} are the angle information respectively associated to the robot (yellow circle) and to the particles (white circles), and are computed by the differences between the heading of the robot/particle (black arrow) and the gradient of the FSD scalar field (red arrow) in the cell corresponding to the robot/particle position. (b) Example of robot's angle information obtained in the local grid map. The red arrow points in the direction of the smallest FSD value. (c) Example of particles' angle information obtained in the global grid map.



(a)



(b)



(c)

From the scalar density field it is also possible to obtain orientation information using standard gradient extraction methods, which we use to compute $f_\alpha(\boldsymbol{p}_t^{[i]})$. In this paper we use the Sobel operator[3], but one can also pre-calculate approximate orientations with methods such as used by Liu et al. (LIU; ZHANG; YOU, 2007). To estimate orientation information using densities, we compute the angle difference, α_{robot}, between the robot orientation, $\theta(\boldsymbol{x}_t)$ and the gradient descent of the FSD scalar field. Then, for every particle, we obtain the difference, α_{part}, between the orientation of the particle and the pre-computed gradient stored in the map. Figure 3.3 illustrates the concepts of α_{robot} and α_{part}, where the black arrows represent the heading of the robot/particles and the red arrows represent the gradient descent of the FSD scalar field in the position of the robot/particles, which always point to the region of smallest density value (usually, near walls). The angle tolerance filter is calculated checking if the difference between α_{robot} and α_{part} is smaller than a threshold ϵ_θ.

$$f_\alpha(\boldsymbol{p}_t^{[i]}) = \begin{cases} 1 & \text{, if } |\alpha_{robot} - \alpha_{part}| \leq \epsilon_\theta \\ 0.5 & \text{, otherwise} \end{cases} \tag{3.5}$$

Equation 3.5 gives less importance to particles with incorrect angle differences. Unfortunately, angle cuts have shown noisy behavior next to and at local maxima, local minima and walls. Therefore, we compute the weight only in regions where the gradient of the densities is stable (i.e. places where the direction of the gradient does not drastically vary between adjacent cells). Taking such precautions, and setting the threshold ϵ_θ to considerably large differences, the results are useful.

Finally, we also can combine different free-space densities with different kernel bandwidths to improve global localization results. The advantage of our method is the computation time – one order of magnitude faster than traditional methods –, which is an important factor for robots with limited processing power. Thus, the combination of a small number of kernels implies small extra online computation time. In order to facilitate the notation, let us rename the weight $w(\boldsymbol{p}_t^{[i]})$ of Equation 3.3 to $w(\boldsymbol{p}_t^{[i]}, K_j)$, i.e. the weight obtained using a given kernel K_j. Then, we can define the weight $w(\boldsymbol{p}_t^{[i]})$ of a particle considering multiple kernel density estimates as just the product of the individual

[3]The Sobel operator is a discrete differentiation operator, which approximates the gradient of the values in a grid (SZELISKI, 2010). Considering the values associated to the 8-neighbors of a cell m_c as $\begin{bmatrix} t_l & t_c & t_r \\ m_l & m_c & m_r \\ b_l & b_c & b_r \end{bmatrix}$, the Sobel operator defines the gradient vector as $\begin{pmatrix} t_r + 2m_r + b_r - t_l - 2m_l - b_l \\ t_l + 2t_c + t_r - b_l - 2b_c - b_r \end{pmatrix}$.

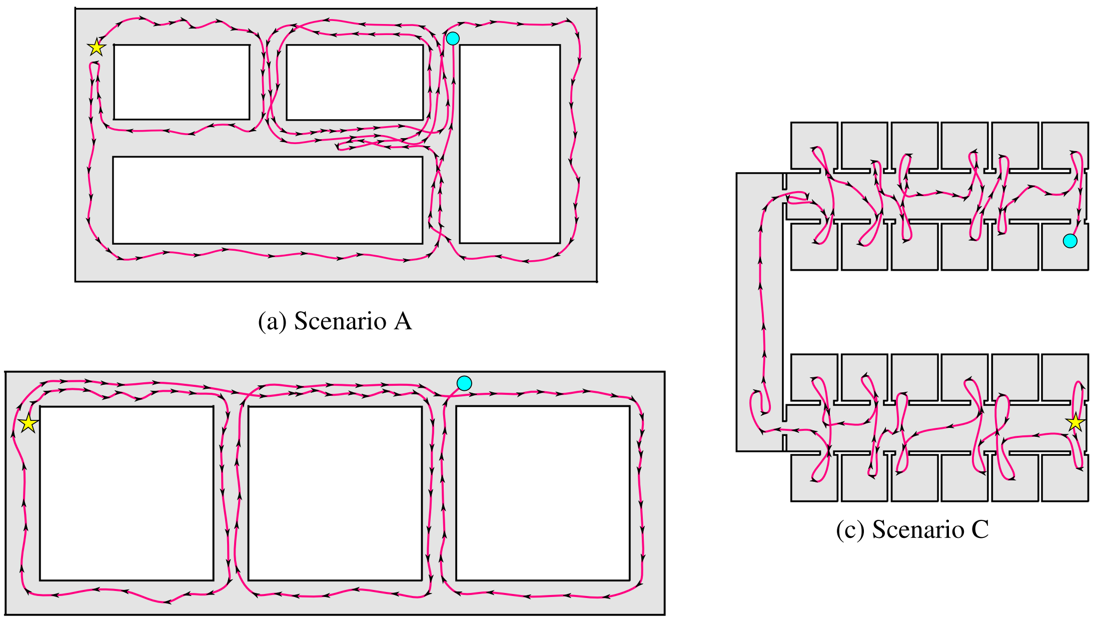
weights, i.e.,

$$w(\boldsymbol{p}_t^{[i]}) = \prod_j w(\boldsymbol{p}_t^{[i]}, K_j). \tag{3.6}$$

3.3.2 Experiments

Our experiments were conducted using the platform described in AppendixA – a Pioneer 3DX robot equipped with a SICK LMS-200 laser range finder, and a notebook with a Intel® QuadCore™ i7 processor with 16GB of RAM memory. The proposed observation model was evaluated in three different simulated scenarios, each one with a specific trajectory (see Fig. 3.4). Scenarios A and B are composed of corridors arranged in adjacent loops, which are highly symmetrical configurations, specially Scenario B. Scenario A contains two small loops of same length ($46m$) and two larger loops of different lengths ($58m$ and $78m$), while scenario B contains three loops of same length ($72m$). Scenario C contains several small rooms in two separate galleries, inside an area of approximately $25m \times 25m$.

Figure 3.4: Scenarios used in the experiments, showing the initial robot position (yellow star), final robot position (blue circle), robot path (pink) and path direction (black arrows).



(a) Scenario A

(b) Scenario B

(c) Scenario C

All maps were discretized using a regular grid with $10\text{x}10cm^2$ cells. In all experiments, we set $r_{max} = 9m$ (radius of the local map), $\epsilon_d = 0.2$ (density threshold) and $\epsilon_\theta =$

$45°$ (orientation threshold)[4]. Tests were divided in two subsections, the first considering only the proposed observation model, and the second showing a comparison with other strategies. For this experiment, we do not pre-compute the gradient angle for each of the stored density maps, but this could also be done, reducing the computation time even further.

3.3.2.1 Evaluating the FSD observation model

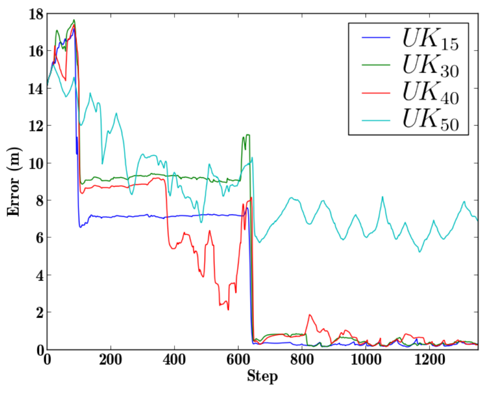
In this section, we present an evaluation of different kernels regarding localization capabilities and computation time in Scenario C (robot path, starting and ending positions are presented in Figure 3.4(c)). Figures 3.5(a)-(c) show the mean localization errors for UK, GK, and IK considering bandwidths of 15, 30, 40, and 50 cells. Figure 3.5(d) presents three approaches, each one combining kernels of same profile and different radius (15, 30 and 50 cells). We performed 15 tests for each configuration, setting the number of particles to 40.000. We did not reset the filter in the case of filter divergence to observe the full behavior of our strategy.

We can see in Figure 3.5 that before the final convergence the errors vary substantially. This happens because the environment is symmetric, thus the particles distribution will be divided in multiple modes. However, in almost all cases the error drops to near zero around iteration 600. Analyzing Scenario C (Figure 3.4(c)), there are two twin galleries connected by a corridor. As the robot leaves the first gallery and enters the corridor, which happens around iteration 600, a mode of the particles distribution hits a wall and vanishes. Afterwards, all particles should ideally converge to the correct solution. However, this does not always happen with large kernels, as can be seen in Figures 3.5(a) and 3.5(b). The problem is that the hard FSD is unreliable when there are unvisited cells inside the kernel (centered in the robot). That said, the high number of gaps and doors in this map results in numerous undefined FSD occurrences when using large kernels, since the observation model erases information of distant cells in order to preserve local consistency of the online constructed map. The best results were presented by the multi-kernel approaches as we can see in Figure 3.5(d). This is expected given that such approaches are able to extract more information from the map.

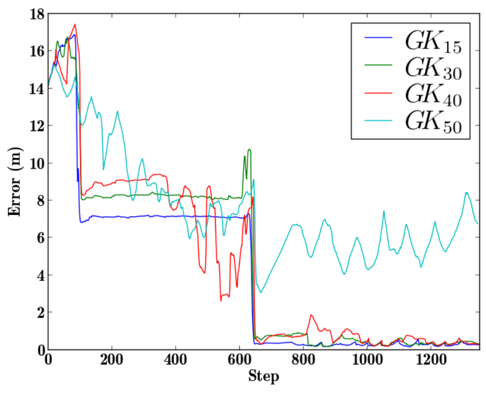
Regarding the computation time, each filter iteration of the approaches using single kernel profiles (Figures 3.5(a), 3.5(b) and 3.5(c)) was performed in around $30ms$. The mean computation time for approaches combining three kernels (Fig. 3.5(d)) was three

[4]Those values were perceived as adequate thresholds in preliminary experiments

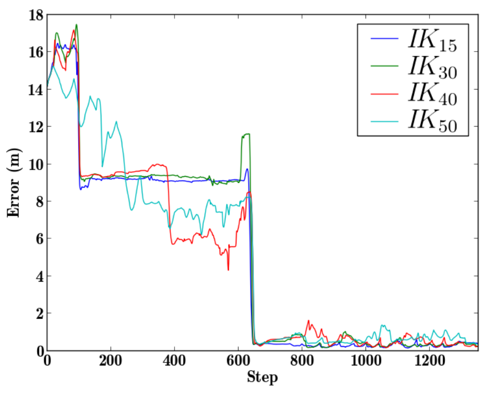
Figure 3.5: Localization in Scenario C using different kernels profiles and bandwidths. (a) Uniform kernel. (b) Gaussian kernel. (c) Inverted kernel. (d) Homogeneous combinations of kernel profiles having different radius – 15, 30 and 50 cells.



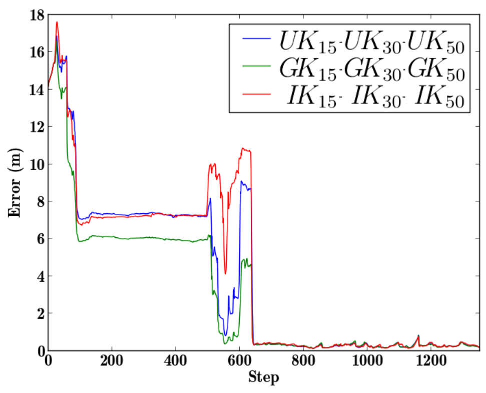
(a)



(b)



(c)



(d)

times larger, around $90ms$.

3.3.2.2 Comparing FSD with other techniques

We compare the proposed observation model with two strategies. The first is an adaptation of the basic beam observation model (which will denoted by **bb**, for short, from now on) described by Fox et al. (FOX; BURGARD; THRUN, 1999). The **bb** method projects each laser beam for each particle using ray-casting and compares to the real sensor observations. To increase performance the authors propose sampling laser beams spaced by $10°$. We do not pre-compute selected orientations as a possible improvement suggested by the authors because such action can lead to an unreasonable amount of memory. For instance, consider that we pre-compute all measurements in a map having 2.000×2.000 cells (the size of our grid), sampling 8 from 181 laser beams and varying the robot orientation at each five degrees. In this case, such approach would require an unfeasible storage space of more than 8GB of memory. In comparison, our method using densities requires the pre-caching of gradients and density estimates for each cell that consume approximately 91MB.

The second strategy is the cosine similarity measure between two sets of readings (which we will denote **cs**). We consider the set of laser beams as a N-dimensional vector, where each row of the vector corresponds to the range measured by a laser beam. Thus, for each particle, we compute the cosine distance between the normalized vector $\mathbf{v}_r$ associated to the robot and the normalized vector $\mathbf{v}_p$ associated to the particle. The idea is the smaller the angle between two vectors the more similar they are. This measure is efficiently computed through the dot product of the vectors.

$$\mathbf{cs}(\mathbf{v}_r, \mathbf{v}_p) = \mathbf{v}_r . \mathbf{v}_p \tag{3.7}$$

However, in practice **cs** presents a slow decay, which implies in setting similar importances for most particles. Thus, we modify Equation 3.7 to obtain a fast decay of the function associated to the particles weights (since all values are normalized, we empirically decided to modify **cs** through exponentiation).

$$\mathbf{fcs}(\mathbf{v}_r, \mathbf{v}_p) = (\mathbf{cs}(\mathbf{v}_r, \mathbf{v}_p))^4 \tag{3.8}$$

In order to compare all techniques, experiments were performed in the Scenarios A, B and C, varying the number of particles. We performed 15 runs in each scenario for each technique and computed the mean error *per* step as well as the mean time *per* iteration step (see Fig. 3.6). In all scenarios, **bb** and **fcs** were compared with FSD using three homogeneous combinations of kernel profiles UK, GK, and IK, with kernel bandwidths of $h = \{15, 30, 50\}$ cells. Since the computational cost of the methods is different, we decided to perform them with different number of particles to allow the methods to run in similar time (between 0.05 and 0.15 seconds per iteration). Both **bb** and **fcs** were tested with 500 particles in Scenario A, 1000 particles in Scenario B and 2000 particles in Scenario C, while our proposed techniques were tested with 20000 particles in Scenario A, 30000 particles in Scenario B and 40000 particles in Scenario C.

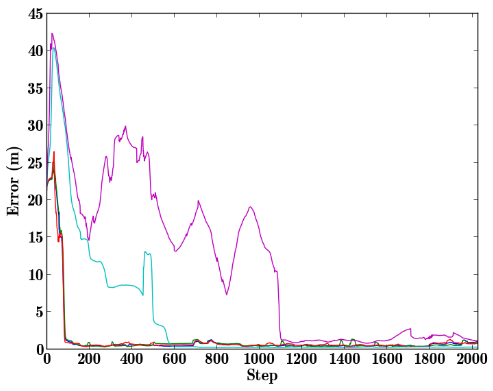
As we can see, in spite of the large differences in the number of particles, we observe in Figures 3.6(b), 3.6(d), 3.6(f) that FSD is considerably faster than the competing methods. Also, as expected, the increase in the number of particles from 500 to 1000 and 2000, implies linear increase in the mean computation time of the competing methods. Thus, substantially increasing the number of particles can be problematic for both competing methods. On the other hand, the proposed observation model does not suffer with such problem. In fact, the change in the number of particles from 20000 to 30000 and 40000, resulted in modest changes in the mean computation time. Note that, even though the observation model can use kernels of considerably large bandwidths and a large amount of particles, the overall computation time is small. The kernel density estimate surrounding the robot is computed only once per step, what makes the kernel size computationally inexpensive when compared with the cost of weighting and resampling all particles according to their pre-computed density values.

Figures 3.6(a), 3.6(c), and 3.6(e) present the mean error per step considering 15 runs of each observation model. In scenario A, all techniques converge to the correct robot position at most after step ≈ 1.100. We can see that the **fcs** method (cyan) converges faster than **bb** (purple), with the exception of Figure 3.6(c) in which both methods do not converge. However, the methods based on free-space density (UK - dark blue; GK - green; and IK - red) converge much faster (around step 100), while using a set of particles 40 times larger than **fcs** and **bb**.

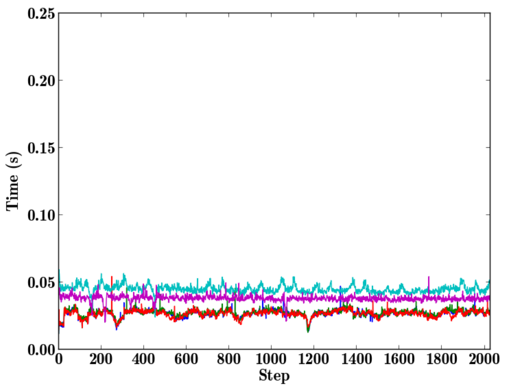
Scenario B, the largest one, renders weak results for the competing methods, meaning that 1.000 particles is probably not enough to resolve the global localization problem in such map. The degree of ambiguity in such scenario easily makes particles diverge. The FSD techniques were the only ones to properly converge to the correct solution, maintaining a computation time about 50% smaller than the competing methods.

The situation is similar for scenario C, where we see **bb** converging to the wrong particle and keeping them until ambiguity is eliminated by the motion model near step 600. The **fcs** method cannot properly converge even at the end of the process. Once again, the results of the proposed method were the ones with the smaller error.

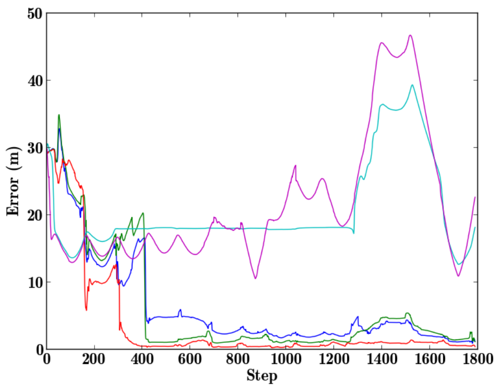
Figure 3.6: Mean errors (a, c and e) and mean times per iteration (b, d and f) for scenarios A, B and C, respectively, displaying: **bb** (purple), **fcs** (cyan), and FSD using three homogeneous combinations of kernel profiles UK (dark blue), GK (green), and IK (red) presented in Section 3.3.2.1 using a set of bandwidths $h = \{15, 30, 50 \text{ cells}\}$. Observe that our technique presents considerably faster times.



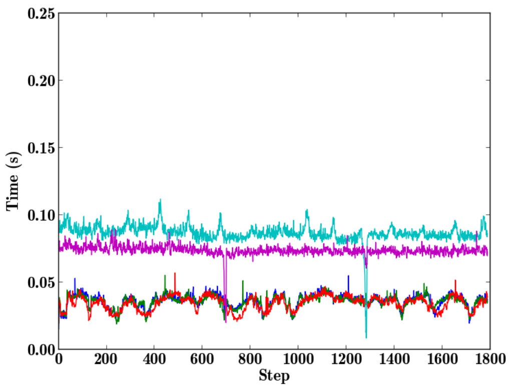
(a)



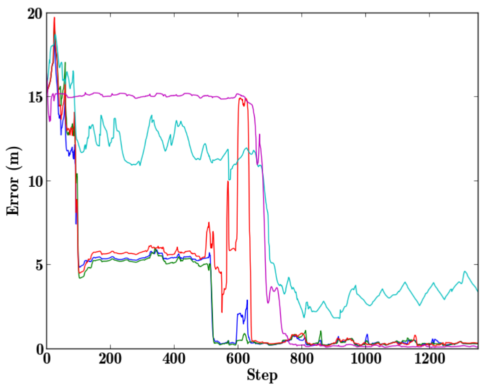
(b)



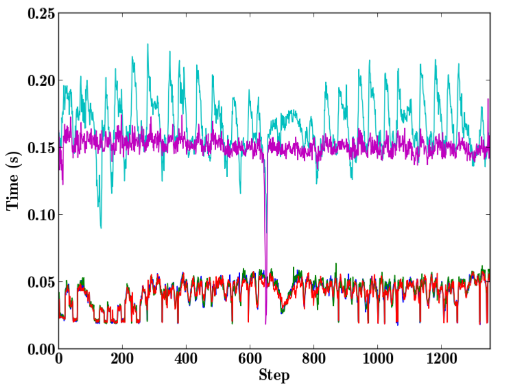
(c)



(d)



(e)



(f)

3.4 Related Work

State estimation problems, such as localization and SLAM, require models that translate sensors observations to a map representation. The so-called observation models are used to evaluate the similarity between true robot observations and observation hypotheses extracted from a specific location given a map of the environment.

Early works relied mostly on feature-based observation models, for instance extracting geometric beacons (e.g. walls, corners) from sonar data (LEONARD; DURRANT-WHYTE, 1991; COX; LEONARD, 1994). Nowadays, feature-based methods are popular due to the use of cameras, making the definition of such models directly linked with the area of computer vision (LOWE, 2004; BAY; TUYTELAARS; GOOL, 2006; CHOI; LEE; OH, 2008). Still, range finders remain highly popular in robotics due to their precision and robustness.

Fox et al. (FOX; BURGARD; THRUN, 1999) propose an observation model for laser range finders, which describes a beam model considering noise in the measurement, errors due to unexpected objects and errors due to failures in object detection. The main problem of such approach is that it requires ray-casting computations, that can be computationally expensive depending on the application. In fact, every time that we want to evaluate an observation from a different pose (x, y, θ) of the map we need to project all scans using ray-casting. If the map is known – for instance in the localization problem – the ray-casting can be pre-computed reducing substantially the time cost. However, such pre-caching drastically increases the memory requirements. Thrun et al. (THRUN, 2001) propose the likelihood field model, which avoids ray-casting by only evaluating the proximity to obstacles of sensor beams endpoints . Their method obtains fast results, but since it only checks the endpoints of a reading, it consider readings that pass through obstacles.

Another way to model sensor observations is with correlation measures. Weiss et al. (WEISS; WETZLER; PUTTKAMER, 1994) propose cross-correlations of orientation and translational histograms of scan readings. Olson (OLSON, 2009; OLSON, 2015) relies on the likelihood field model to propose efficient multi-level strategies for correlative scan-matching. Duckett and Nehmzow (DUCKETT; NEHMZOW, 2001) present a technique which stores a pair of histograms of free cells and obstacles over a discrete grid obtained by measurements taken at a specific orientation. They use a robot equipped with a compass, which is used to ensure that the robot is at that same orientation when performing measurements. Matching is performed convolving the two histograms obtained by the robot with every pair of histograms previously calculated and stored in each cell of the map.While their results are good, the assumption about the compass and the ability to know the precise robot orientation is a major drawback of their method.

Lastly, the more compact form of representing observations is translating them to a single value. Zhang et al. proposes a single-valued observation model called Similar Energy Region (SER) (ZHANG; ZAPATA; LÉPINAY, 2012). SER is a value associated to each position in the free-space, corresponding to the sum of the ranges of all readings made by the robot at such position. Since they use a robot able to get readings in $360°$, their measure of energy is virtually independent of orientation (this is only true under the assumption of a perfectly circular robot, with uniformly distributed sensors). A problem of such strategy is that the simple sum of measured ranges can easily produce misleading values, i.e., similar results on very different regions.

Our proposal is similar to the concept of SER, but we avoid misleading values by constraining our measurement to a local circular region. Instead of obtaining an absolute measurement of free space surrounding the robot, we compute a kernel density estimate (KDE) that returns a free-space ratio in terms of the maximum area of the local region.

3.5 Summary

In this chapter we have proposed the Free-Space Density (FSD) which is a novel observation model based on kernel density estimates of the free-space cells surrounding the robot. We have evaluated FSD in the problem of global localization of a mobile robot using particle filters. As the experiments have shown, the computation time of our technique is remarkably low in comparison with other techniques, but obtaining similar and better quality of results.

The main advantages of our proposal are:

(i) Conciseness – Each position in space is associated to a single value;

(ii) Robustness – Small variations or noise in the robot position do not strongly impact on the FSD, because the variation of the density values is always smooth, as illustrated in Figure 3.2. Note that the FSD smoothness increases with the kernel size.

(iii) Estimation of the robot orientation using the gradient of the scalar field of FSDs, as described in Section 3.3.1;

(iv) Low memory consumption – Differently from approaches such as Fox et al. (FOX; BURGARD; THRUN, 1999), in which pre-caching the measurements estimates adds a considerable memory burden to the process, the pre-caching of FSD implies the addition of only one value per cell of the grid (see Section 3.3.2.2);

(v) Low time cost – The computational time of measuring any position in a pre-computed map correspond to a single-valued table look-up.

Note that the proposed method performs a dimensionality reduction of the information. While this allows all the benefits mentioned above, we highlight that it decreases its precision and effectiveness. If we use a small number of particles to localize the robot, the method will most certainly be unsuccessful. Fortunately, we can do a massive increase in the number of samples, with minimum impact on processing time.

Our method also depends on known cells surrounding the robot. For instance, when the robot is approaching corners and bifurcations, the method is unable to establish a valid observation until the current local map is complete – at least in the surroundings of the robot. If we always use the current local map, correspondence problems may arise. For example, looking back at Figure 3.1, and supposing that the robot is going upwards and not downwards (i.e., from the position in (b) to the position in (a)). Then, part of the yellow area shown in Figure 3.1(a), which contributes to the computation of the kernel density estimate, will be unobservable until the robot actually enters the block of rooms. This may cause large disparities in the value of the soft FSD (Ψ), while the hard FSD stays undefined ($\Psi^{\diamond} = UND_{\Psi}$). To circumvent this problem, our localization method avoids refining the state estimate in such situation, thus, increasing the uncertainty of the filter.

The aforementioned characteristics make our method better for global localization than tracking. In particular, it is suited for global localization in large maps, where a large number of hypothesis must be tested at once. It is possible to combine the proposed observation model with others to achieve better localization results once the rough position of the robot is found – e.g. changing to a more accurate and computationally expensive observation model with less particles.

4 N-GRAM SLAM: TRANSLATING SIMPLE OBSERVATION VALUES INTO WORDS AND SOLVING SLAM WITH A SHALLOW LINGUISTIC PROCESSING TECHNIQUE

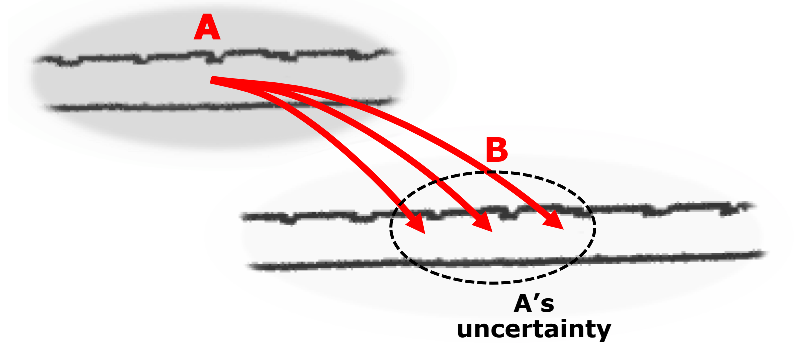
4.1 Introduction

A critical part of simultaneous localization and mapping (SLAM) is its front-end, that is, to solve the place recognition problem. Even if we use the best optimization algorithm available in the literature as SLAM back-end, the resulting map will be poor if the front-end is not good. Nonetheless, while wrong place recognition can be disastrous for the SLAM process, proper place recognition provides essential information for the correction of localization and mapping errors (THRUN; BURGARD; FOX, 2005; STACHNISS; HAHNEL; BURGARD, 2004; GRISETTI et al., 2010b). It is important to store observations made by a robot during its trajectory in order to be able to match regions observed in different moments, which means to detect when the robot is revisiting an already known place (WERNER et al., 2009; MILFORD; WYETH, 2012).

There are basically two criteria for a precise place recognition (OLSON, 2009). The first criterion – *global sufficiency* – ensures that the matched region is large enough to be unambiguous, that is, distinct from all other regions where the robot can be located considering the robot uncertainty. If the robot has large uncertainty about its location, then the search space for matches is also large, possibly containing many ambiguous places. Thus, the larger the matched region inside the search space, the smaller will be the chances of obtaining a false matching. The second criterion – *local uniqueness* – ensures that the matched region is unique inside its surrounding area, meaning that the match is probably correct, otherwise the position uncertainty of the robot is extremely wrong. Most methods focus on finding and matching highly distinguishable features in the environment to ensure local uniqueness. However, in many situations, depending on the ambiguity level of the environment and the sensors used, it is very hard to fulfill such criterion. This happens in the so-called *picket fence* problem, illustrated in Figure 4.1: many possible good matches are nearby and can be obtained by simple translations, like skipping fence posts of a picket.

The core assumptions in our work are that the robot equipped with a laser range-

Figure 4.1: Example of the picket fence problem in an environment. Multiple local ambiguities can lead to wrong place recognition. Figure adapted from (OLSON, 2009)



finder moving inside a structured indoor environment does not know where it is; it has no information about the map; and the uncertainty of its odometry is high. Additionally, the robot may perform multiple runs starting from different initial points, i.e. we may have to try matching different trajectories. In such situation, local uniqueness is extremely hard to obtain, therefore the only solution is to find matches of long regions. However, comparing long sequences of raw sensor readings not only can be costly, but also suffer in the presence of noise.

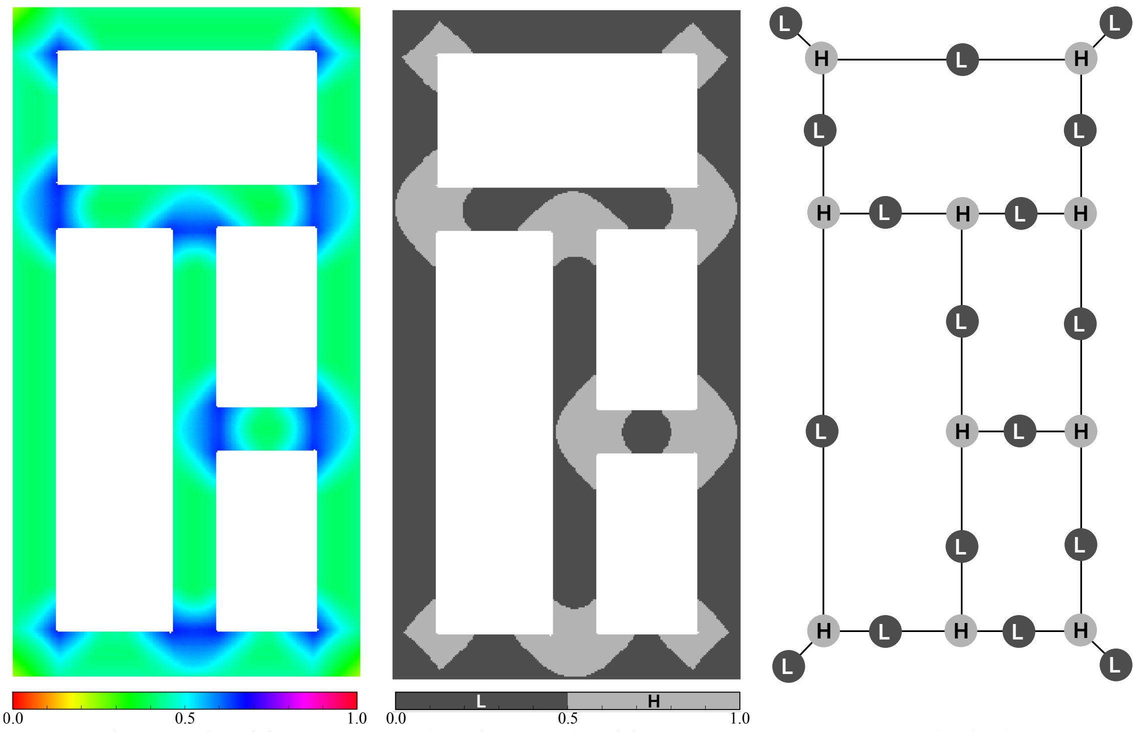
Our proposal[1] is to view this problem as a prediction and correction problem of information transmitted through a noisy channel, which has been widely studied in fields like computational linguistics. Defining a robust and compact text description for the sequences of sensor readings allows us to efficiently handle the place recognition problem with the aid of techniques from linguistic processing, such as n-grams. That said, the novel observation model described in last chapter, the Free-Space Density (FSD), performs a large compression of the information measured by the robot at each time step: all sensor data is transformed into a single value. Additionally, some of the most valuable features of FSD are stability and robustness. The variation of free-space density measures is smooth, that is, we do not get abrupt density changes in an environment (except if using really small kernels, which is useless in general). This way we can adopt FSD to easily classify each point of the environment and generate coherent regions.

In Figure 4.2(a), we present a map of densities obtained using the inverted kernel with bandwidth of 30 cells, as shown earlier in Figure 3.2. As we can see, when the free-space density varies, such variations gradually go from light green ($\Psi^{\diamond} \approx 0.4$) to dark

[1]The core idea of this chapter was originally published in the paper "Using n-grams of spatial densities to construct maps" (MAFFEI et al., 2015b), presented at the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'15).

blue ($\Psi^{\diamond} \approx 0.7$) and vice-versa. Conversely, the densities stay constant in large uniform regions of the environment, e.g. corridors. Thus, if we discretize the density values into a small number of classes we are able to divide the environment into contiguous regions of homogeneous density. As shown in Figure 4.2(b), the simple segmentation of densities into two classes, such as low ($\Psi^{\diamond} \leq 0.5$) and high ($\Psi^{\diamond} > 0.5$), allows the separation of corridors from corners and bifurcations. Furthermore, a topological representation of the environment can be easily built considering this information, as illustrated in Figure 4.2(c).

Figure 4.2: Example of a topological map obtained from a densities map by just applying a threshold over density values. The original densities, in (a), are divided into two classes, in (b), resulting in two types of contiguous regions represented in the graph, in (c).



(a) Continuous densities map (b) Discrete densities map (c) Topological map

The segmentation of the FSD space is a way to translate relevant spatial information of the environment into a very compact representation. We can compress large sets of raw laser readings into sequences of few density values. Such information can be described by simple words. Place recognition is performed matching ordered sequences of n words, also denominated n-grams, reaching maximum precision for some scenarios. We apply our front-end on a graph-based SLAM framework using a traditional Gauss-Newton back-end for graph optimization, previously described in Section 2.1.3.

We begin this chapter presenting a review on n-grams, which is a simple technique used in shallow linguistic processing for word prediction, matching of texts, among other applications. Later, we detail the main steps of the proposed SLAM strategy: the translation of density signatures into words, the matching of n-grams, and the topological map construction. The evaluation of our method is made through experiments in different environments and sets of configurations. Finally, we draw some conclusions about the proposed algorithm.

4.2 SLAM using n-grams of FSD-based words

Our SLAM approach is based on the idea that spatial regions have aspects, e.g. free-space density, that remain fairly consistent even when being traversed using different paths. By translating such aspects into words we reduce the search space for place recognition. We also analyze the ordered sequence of words, i.e. n-grams, to efficiently disambiguate similar regions.

The general algorithm that we developed for place recognition is presented in Algorithm 4.1. Each step is following described in details.

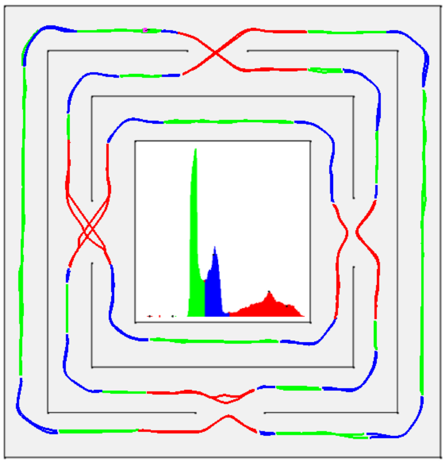
Algorithm 4.1: Basic N-Gram SLAM algorithm

```
  while exploring do
1     Build contiguous spatial region.
2     Translate region into word.
3     Match current sequence of words with past words using $n$-grams.
4     Update the topological map.
```

4.2.1 Building a contiguous spatial region

The first step of the algorithm is to obtain a contiguous region of similar free-space density (FSD). At each robot pose, we compute the FSD as described by Algorithm 3.1 in Chapter 3. Then, the density values are discretized into few significant density classes. This is fundamental not only to reduce noise effects, but also to posteriorly obtaining a low-dimensional space of words. We can just uniformly quantize the density values into classes of same interval range, or divide the classes more properly considering the history of observations. For this, we use the Mean Shift algorithm over the histogram of all past observations (CHENG, 1995). Figure 4.3 shows an example of density classification using Mean Shift for a given robot path. The histogram, depicted in the center of the image,

Figure 4.3: Result of the density quantization using Mean Shift. The algorithm detected three peaks in the histogram of densities, as shown in the center of the image. The densities obtained in each point of the robot's path are depicted using the color associated with the corresponding density quantization.

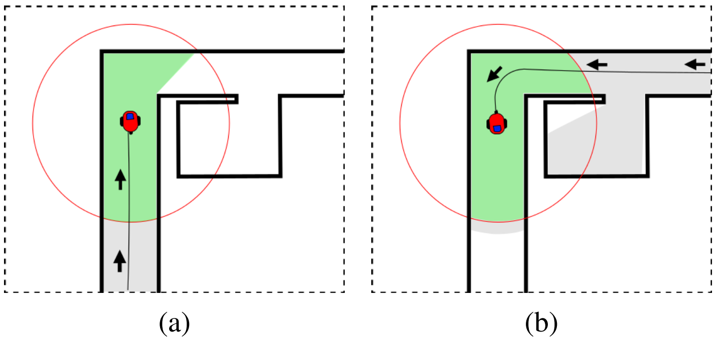


presents three maxima in the colors green, blue and red, which represent, respectively, the average densities of corridors, corners and crossroads/bifurcations. However, the distribution of densities may change over time due to new areas being explored. To deal with such problem, the Mean Shift can be periodically updated. If it results in a drastically different partition of density classes, the words corresponding to all previous observations must be regenerated considering the new density partitions.

An important point of the algorithm is that, instead of using the *hard FSD* ($\Psi^{\diamond}$), we use the *soft FSD* (Ψ), which always computes a valid density, even when unknown map cells are found nearby the robot. Figure 4.4 illustrates a situation in which the densities assume different values in the same point because of variations in the direction of the robot motion. When the robot is going up the vertical corridor just before turning right at the corner, as shown in (a), there are unknown cells inside the kernel that are reachable from the robot position. In this case, the hard FSD is undefined, while the soft FSD corresponds to an unreliable value, i.e. only occurs when the robot moves in that specific direction. In contrast, when the robot is moving in the reversed direction, as shown in (b), all reachable cells inside the kernel are known (either obstacles or free-space), therefore, the values of soft and hard FSD are the same.

Given that situations like the presented in Figure 4.4 are quite common, the problem of using hard FSD is that many times the density values are undefined. Note that there are no points where the hard FSD will be always undefined, in fact, this is something that depends on the robot path. This can be observed in Figure 4.5, in which we present soft and hard FSD values computed in segments of robot paths varying the motion direction. The hard FSD is undefined in large part of the paths, as shown in (a) and (b), while the soft FSD is always valid, as shown in (c) and (d). Furthermore, if the robot is revisiting a place with unknown areas (e.g. a corridor with doors to unexplored rooms) in the same direction, the method will obtain the same sequence of soft FSD values.

Figure 4.4: Different motion directions lead to different FSD values because the area observed by the robot changes. (a) Unreliable FSD value, i.e. $\Psi \neq \Psi^\diamond$, because reachable unknown cells inside the kernel lead to $\Psi^\diamond = UND_\Psi$. (b) Reliable FSD value, i.e. $\Psi = \Psi^\diamond$. *Description*: kernel boundaries (red circle), known area (gray region), known area inside the kernel that is reachable from the robot position (green region), motion direction (black arrows).


(a) (b)

Observations made by the robot are grouped in a list $O = [\boldsymbol{o}_0, \cdots, \boldsymbol{o}_t]$, where

$$\boldsymbol{o}_t = \langle \Psi, d, \boldsymbol{x}_t \rangle \tag{4.1}$$

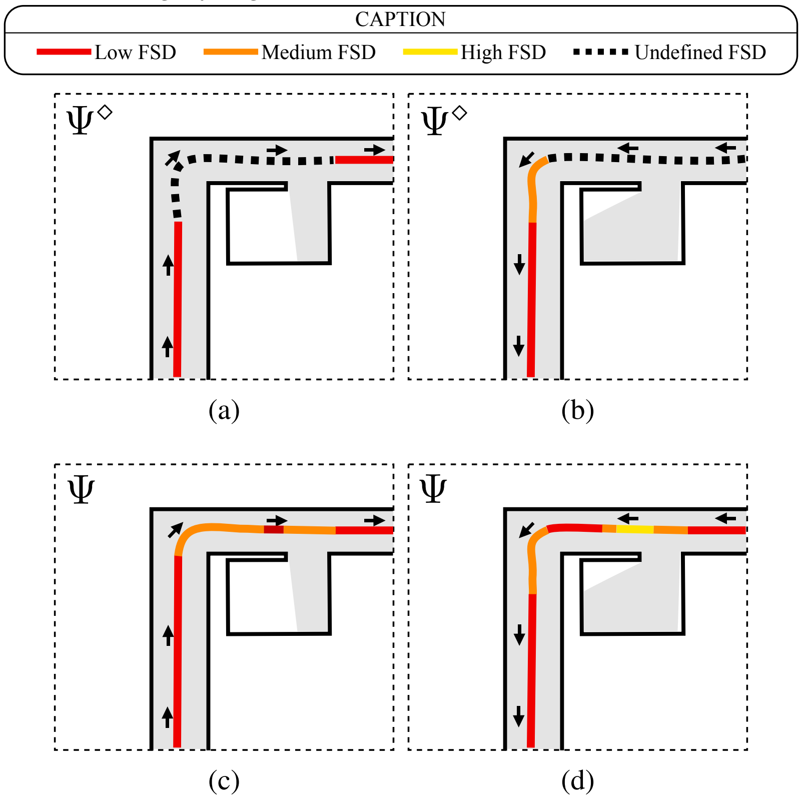
is the observation made by the robot at instant t and it is composed of a density value (Ψ) with its associated density class (d), and the robot pose ($\boldsymbol{x}_t$) given by odometry at instant t. Our method divides O into smaller lists of observations representing contiguous regions, that is, a region R_n is the n-th sequence of observations in O that share the same density class. We can rewrite O as a concatenation[2] of all built regions, followed by a list P of "pending observations", which correspond to the last observations made by the robot that are still not associated to any region,

$$O = R_0 \oplus R_1 \oplus \cdots \oplus R_{n-1} \oplus R_n \oplus P. \tag{4.2}$$

The algorithm for building contiguous regions, by clustering sequential observations of similar densities, is described in Algorithm 4.2 and its input is the list of pending observations. The algorithm is executed whenever a variation in the density class occurs. In

[2]We define the symbol $\oplus$ as the operator to concatenate lists.

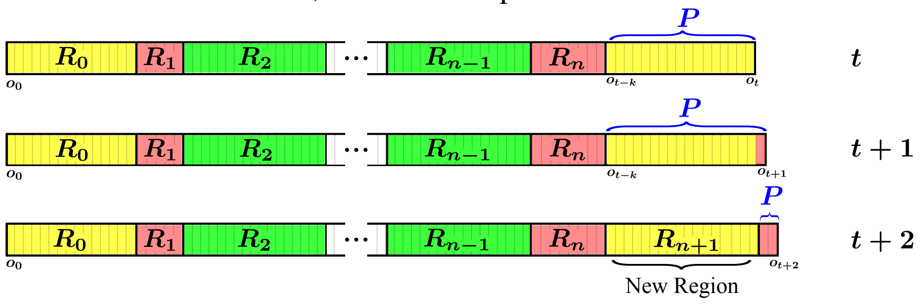
Figure 4.5: Differences on the results of soft and hard FSD due to variations in the robot motion direction. (a)-(b) Regarding hard FSD, there are large segments of undefined density value. (c)-(d) Regarding soft FSD, there are discrepancies near corners and in the region near the door to the room, which considers the known area inside the room (larger when the robot is moving to the left). *Description*: FSD values in the robot path (colored thick lines), known area (gray region), motion direction (black arrows).



order to exclude wrong observations resulting from noise or temporary partial occlusions (e.g. fast moving obstacles), we set a threshold (t_o) as the minimum size of a valid region. Thus, if the size of P is smaller than t_o, no region is generated (lines 2-3). If this is not the case, then a region is created containing the sequence of all observations that shares the same density class (d_{new}) of the initial observation (lines 4-8). The output of the algorithm is the new contiguous region (R_{new}), along with the remaining of the list of pending observations (P). When we are able to successfully generate a new region, R_{new}, we must replace the original P (used as input for the algorithm) in O, by the concatenation of R_{new} and the updated P.

Figure 4.6 exemplifies a few steps of the algorithm. At instant t, the last built region is represented by the observations in R_n, that ends before $\boldsymbol{o}_{t-k}$. Therefore, the list P contains the last k observations made by the robot. At instant $t+1$, a variation in the density class occurs (as illustrated by the difference of colors), causing the generation of a new region. At instant $t+2$, the list P is updated to start with the observation that

Figure 4.6: Example of the algorithm for building contiguous regions. The density class of each region is represented by a different color. The algorithm's input is the list P of the last observations that are not associated to any region, which at instant t corresponds to $[\boldsymbol{o}_{t-k}, \cdots, \boldsymbol{o}_t]$. A region is built at every transition of density class, which, in the example, occurs at instant $t+1$. After this, the list P is updated to start at the observation $\boldsymbol{o}_{t+1}$.


P
R_0 R_1 R_2 $\cdots$ R_{n-1} R_n
o_0 o_{t-k} o_t
t
P
R_0 R_1 R_2 $\cdots$ R_{n-1} R_n
o_0 o_{t-k} o_{t+1}
$t+1$
P
R_0 R_1 R_2 $\cdots$ R_{n-1} R_n R_{n+1}
o_0 o_{t+2}
New Region
$t+2$

follows the new built region.

Algorithm 4.2: Build contiguous region

```
   Input: $P = [\boldsymbol{o}_{t-k}, \boldsymbol{o}_{t-k+1}, \cdots, \boldsymbol{o}_{t-1}, \boldsymbol{o}_t]$
   Output: $R_{new}, P$
 1 $R_{new} = [\ ]$
 2 if $size(P) \leq t_o$ then
 3     return $R_{new}, P$
 4 $\boldsymbol{o} = pop\_front(P)$
 5 $d_{new} = d(\boldsymbol{o})$
 6 while $d(\boldsymbol{o}) == d_{new}$ or $size(R_{new}) \leq t_o$ do
 7     $push\_back(\boldsymbol{o}, R_{new})$
 8     $\boldsymbol{o} = pop\_front(P)$
 9 $push\_back(\boldsymbol{o}, P)$
10 return $R_{new}, P$
```

4.2.2 Translating regions into words

Each contiguous region is translated into a word composed of three syllables,

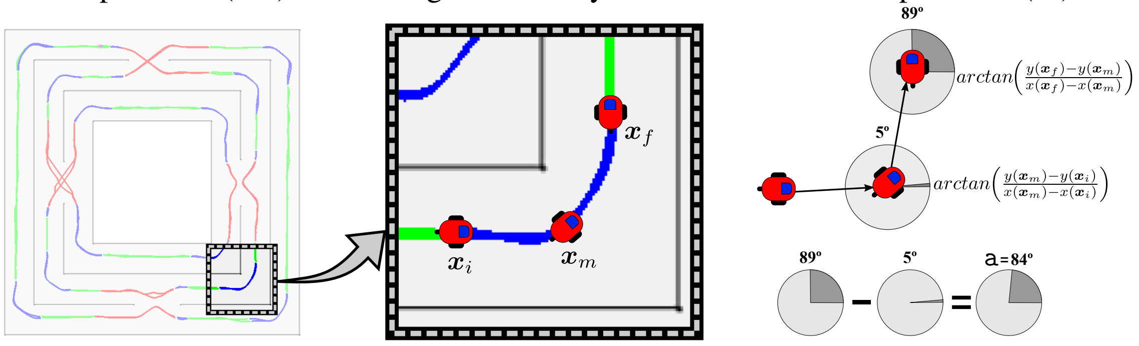
$$\mathtt{W} = \langle \mathtt{d}, \mathtt{s}, \mathtt{a} \rangle, \tag{4.3}$$

that corresponds to information of free-space density, size and angle variation, respectively. Algorithm 4.3 shows the process of word building. The list of consecutive similar observations that compose the region, R, is passed as input to the algorithm. The first syllable of the word, d, represents the density class of all observations in R, which is the density class of the initial observation, $d(\boldsymbol{o}_i)$ (line 1). The second syllable, s, is the size of the region, given by the number of observations (line 2). The last syllable, a, is the difference of the angle between median and final poses of the sequence of observations, and the angle between initial and median poses (line 3). Figure 4.7 shows an example of the angle variation in a segment of the robot path. In the example the robot turns to the left in a corner, which results in an angle of approximately 90 degrees.

Algorithm 4.3: Build Word

```
    Input: $R = [\boldsymbol{o}_i, \cdots, \boldsymbol{o}_{m=(i+f)/2}, \cdots, \boldsymbol{o}_f]$
    Output: W
1   d $= d(\boldsymbol{o}_i)$
2   s $= size(R)$
3   a $= arctan\left(\frac{y(\boldsymbol{x}_f)-y(\boldsymbol{x}_m)}{x(\boldsymbol{x}_f)-x(\boldsymbol{x}_m)}\right) - arctan\left(\frac{y(\boldsymbol{x}_m)-y(\boldsymbol{x}_i)}{x(\boldsymbol{x}_m)-x(\boldsymbol{x}_i)}\right)$
4   W $= \langle$d, s, a$\rangle$
5   return W
```

Figure 4.7: Computing the angle variation of a region. (a) Segment of the robot path. (b) Highlighting the poses of initial, median, and final observations, i.e. $\boldsymbol{x}_i$, $\boldsymbol{x}_m$ and $\boldsymbol{x}_f$. (c) The angle variation (a $= 84°$) is the difference between the angle formed by the final and median positions (89°) and the angle formed by the median and initial positions (5°).


$\boldsymbol{x}_f$
$\boldsymbol{x}_i$
$\boldsymbol{x}_m$
89°
$arctan\left(\frac{y(\boldsymbol{x}_f)-y(\boldsymbol{x}_m)}{x(\boldsymbol{x}_f)-x(\boldsymbol{x}_m)}\right)$
5°
$arctan\left(\frac{y(\boldsymbol{x}_m)-y(\boldsymbol{x}_i)}{x(\boldsymbol{x}_m)-x(\boldsymbol{x}_i)}\right)$
89°
5°
a=84°
-
=

(a) (b) (c)

4.2.3 Matching current and past observations using n-grams

Place recognition is performed by finding matches of n-grams, i.e. sequence of n words, which is a technique widely used in shallow linguistic processing[3]. At each instant, we compare the most recent n-gram with all possible n-grams over the entire path.

[3]See Section 2.3 of Chapter 2 for a detailed background on the n-grams technique.

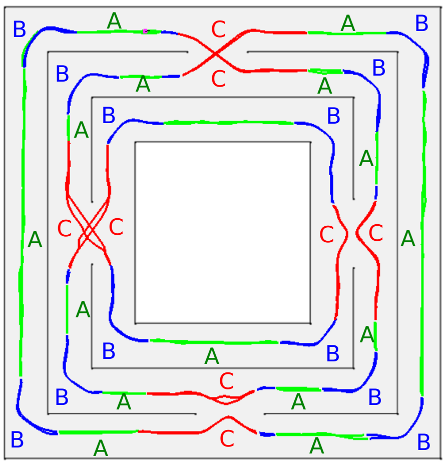
Figure 4.8 shows an example of construction of words and n-grams. In (a), we can see the densities quantization, which separates the environment in three different types of regions – corridors, corners and bifurcations. Three consecutive words are presented in (b), along with their descriptions in (c). For instance, the first highlighted word $\{B, 18, 90\}$ corresponds to a segment of density class B, composed of 18 observations, and an angle variation of $90°$ (i.e. a left turn). Figures 4.8(d)-(e) show, respectively, the 2-grams and the 3-gram that can be originated from the sequence of those words.

A match between two n-grams is considered successful only if there are matches between all words of both n-grams. In the same way, the matching between two words considers all three syllables $\langle \mathtt{d}, \mathtt{s}, \mathtt{a} \rangle$. The density check is binary, i.e. either the words are in the same density class or they are not. The matching of the other two syllables have pre-defined tolerances. The size matching requires the computation of a ratio, $\rho_{\mathtt{s}}$, between the size differences of the two words being compared and the size of the largest word,

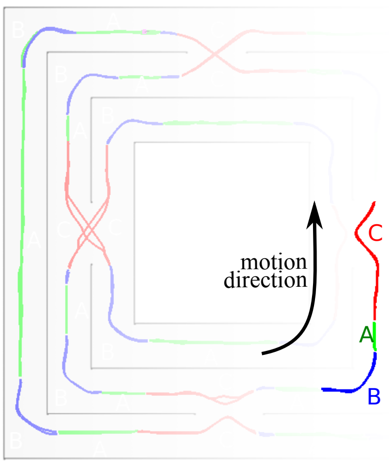
$$\rho_{\mathtt{s}}(\mathtt{W}_i, \mathtt{W}_j) = \frac{|\mathtt{s}(\mathtt{W}_i) - \mathtt{s}(\mathtt{W}_j)|}{\max(\mathtt{s}(\mathtt{W}_i), \mathtt{s}(\mathtt{W}_j))}. \tag{4.4}$$

We accept matches when $\rho_{\mathtt{s}}$ is smaller than a given threshold $\epsilon_{\mathtt{s}}$ (e.g 20%). The angle

Figure 4.8: Example of construction of words and n-grams. (a) Density quantization into three classes. (b) Highlighting three consecutive regions visited by the robot. (c) Description of the words associated to the three regions. (d) Description of the two possible 2-grams. (e) Description of the possible 3-gram.



(a)



(b)

Words:
⟨B,18,90⟩
⟨A,7,0⟩
⟨C,23,5⟩

(c)

2-Grams:
⟨B,18,90⟩-⟨A,7,0⟩
⟨A,7,0⟩-⟨C,23,5⟩

(d)

3-Grams:
⟨B,18,90⟩-⟨A,7,0⟩-⟨C,23,5⟩

(e)

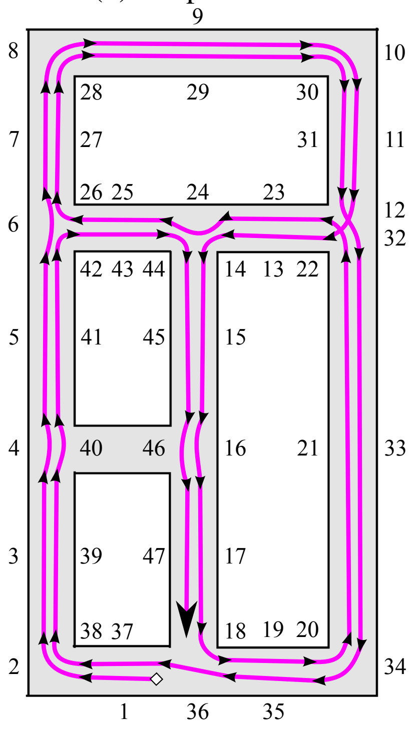
tolerance, $\epsilon_{\mathtt{a}}$, is set to 30 degrees, a value that we empirically determined as significant in our experiments for differing typical robot subpaths such as turning right or going straight ahead.

Figure 4.9 shows a complete illustration of the place recognition strategy. In (a), we have a robot path that is divided into 47 segments of homogeneous densities and crosses every region of the environment more than once.[4] In (b), it is possible to see that, due to the structured nature of this environment (and due to the density quantization into three classes), there are only 10 different types of words that can be generated, which are: turning to the left and to the right at corners ($\mathtt{W}_1$, $\mathtt{W}_2$) and bifurcations ($\mathtt{W}_3$,$\mathtt{W}_4$); going straight in bifurcations ($\mathtt{W}_5$); or going straight in corridors, which have five different sizes ($\mathtt{W}_6$ to $\mathtt{W}_{10}$). In (c), the 47 segments of the robot path are translated into words. This allows searching for n-grams matches with extreme efficiency, because all history of observations made by the robot is now compressed into a few hundreds of bytes. In (d), we show all matches of n-grams that happen with this robot path. When using 2-grams, there are 27 matches (of 13 different types of 2-grams), but only 10 of those matches are indeed correct. When using 3-grams, the total number of matches drops to 12, from which 7 are correct. When using 4-grams, there are 4 matches, all of them correct. Finally, the largest n-grams that still obtain matches are 5-grams, with 2 correct matches. Note that we do not match words that occur in the same place but in reverse directions, like words number 20 (of the type $\mathtt{W}_1$, i.e. a left turn) and number 34 (of the type $\mathtt{W}_2$, i.e. a right turn).

As the example demonstrates, the choice of n to be matched is very important. Using n-grams of small sizes, such as two or three, will lead to a large number of incorrect matches, because such sequences are hardly unique to some path of the environment, specially in ambiguous structured scenarios like the one in Figure 4.9(a). On the other hand, large values of n lead to better matches, since the presence of a same large sequence in different paths through the environment requires the existence of a highly symmetrical environment. However, choosing extremely large n sizes can lead to zero matches.

[4]Note that paths running over the same place are drawn with a significant displacement between them, which could lead to different density values. However, this was done in the illustration just for ease of understanding.

Figure 4.9: Example of place recognition using n-grams. (a) Robot path (starting at white diamond and ending at large arrow) is divided into 47 segments (b) All segments can be described using 10 different possible words. (c) Robot path translated to a sequence of 47 words. (d) All possible matches of n-grams (there are no matches with $n > 5$).



(a)

Possible words in this environment	
W_1: left turn at corner	W_2: right turn at corner
W_3: left turn at bifurcation	W_4: right turn at bifurcation
W_5: straight on bifurcation	W_6: straight on very short corridor
W_7: straight on short corridor	W_8: straight on medium corridor
W_9: straight on long corridor	W_{10}: straight on very long corridor

(b)

Word associated with each segment of the robot path

1	2	3	4	5	6	7	8	9	10
W_6	W_2	W_8	W_5	W_8	W_5	W_7	W_2	W_9	W_2
11	12	13	14	15	16	17	18	19	20
W_7	W_4	W_6	W_3	W_8	W_5	W_8	W_3	W_6	W_1
21	22	23	24	25	26	27	28	29	30
W_{10}	W_3	W_6	W_5	W_6	W_4	W_7	W_2	W_9	W_2
31	32	33	34	35	36	37	38	39	40
W_7	W_5	W_{10}	W_2	W_6	W_5	W_6	W_2	W_8	W_5
41	42	43	44	45	46	47			
W_8	W_4	W_6	W_4	W_8	W_5	W_8			

(c)

All matches of n-grams

2-grams

Total Matches: 27 Correct: 10 Wrong: 17 Types: 13

W_6-W_2 (1-2;37-38); W_2-W_8 (2-3;38-39); W_8-W_5 (3-4;5-6;15-16;39-40;45-46)
W_5-W_8 (4-5;16-17;40-41;46-47); W_7-W_2 (7-8;27-28); W_2-W_9 (8-9;28-29)
W_9-W_2 (9-10;29-30); W_2-W_7 (10-11;30-31); W_4-W_6 (12-13;42-43); W_3-W_6 (18-19;22-23)
W_6-W_5 (23-24;35-36); W_5-W_6 (24-25;36-37); W_6-W_4 (25-26;43-44)

3-grams

Total Matches: 12 Correct: 7 Wrong: 5 Types: 7

W_6-W_2-W_8 (1-3;37-39); W_2-W_8-W_5 (2-4;38-40)
W_8-W_5-W_8 (3-5;15-17;39-41;45-47); W_7-W_2-W_9 (7-9;27-29)
W_2-W_9-W_2 (8-10;28-30); W_9-W_2-W_7 (9-11;29-31); W_6-W_5-W_6 (23-25;35-37)

4-grams

Total Matches: 4 Correct: 4 Wrong: 0 Types: 4

W_6-W_2-W_8-W_5 (1-4;37-40); W_2-W_8-W_5-W_8 (2-5;38-41)
W_7-W_2-W_9-W_2 (7-10;27-30); W_2-W_9-W_2-W_7 (8-11;28-31)

5-grams

Total Matches: 2 Correct: 2 Wrong: 0 Types: 2

W_6-W_2-W_8-W_5-W_8 (1-5;37-41); W_7-W_2-W_9-W_2-W_7 (7-11;27-31)

(d)

4.2.4 Topological Map Construction

We use the graph-SLAM back-end discussed in Section 2.1.3, which performs the Gauss-Newton graph optimization to build the topological map of the environment. Regarding the front-end, our method generates one node for each region, where its pose is set as the pose of the median observation of each word. Edges connect nodes representing adjacent words, as well as those associated by the proposed matching strategy. The observation associated to the first type of edge corresponds to the odometry difference, while the observation associated to the second type is set to 0, because nodes that are matched are ideally the same. Both types of edge use a fixed information matrix. If the densities quantization changes due to an update of Mean Shift, the back-end is set to reconstruct the graph from scratch to accommodate such changes.

4.3 Experiments

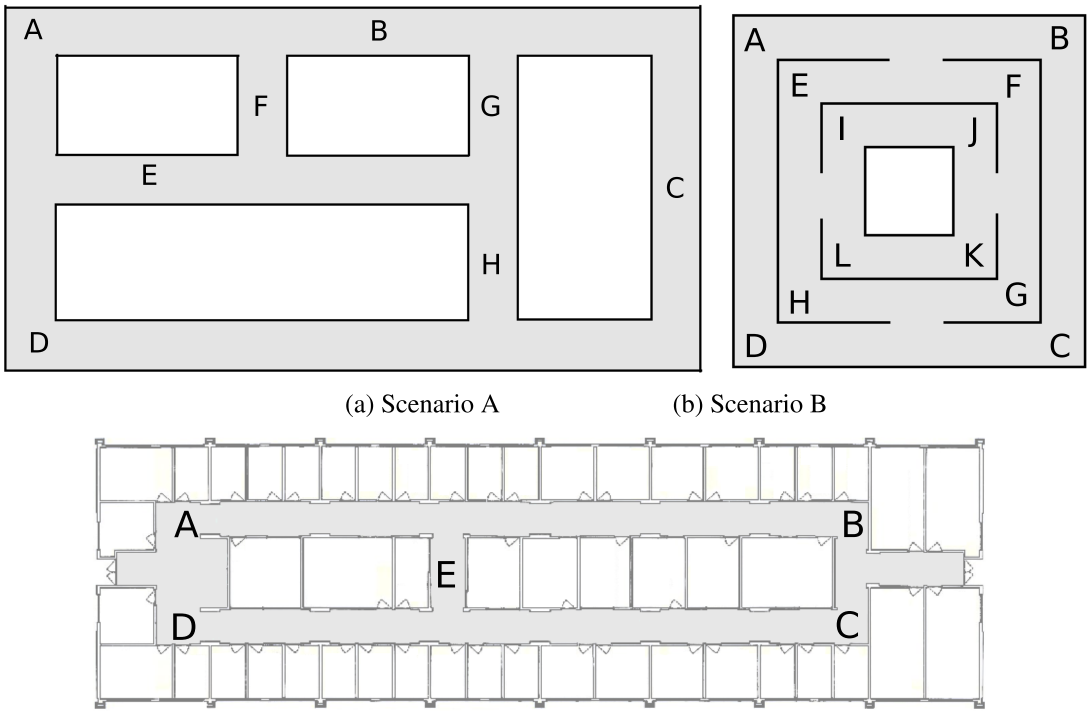
4.3.1 Evaluation Scenarios

Our strategy was evaluated through experiments using the platform described in AppendixA – a Pioneer 3DX robot equipped with a SICK LMS-200 laser range finder, and a notebook with a Intel® QuadCore™ i7 processor with 16GB of RAM memory. Figure 4.10 presents the three scenarios where the experiments were performed: simulated scenario A has four adjacent loops with lengths varying from $44m$ to $78m$; simulated scenario B has three nested loops – the largest has $88m$ of length; and the real scenario C has two adjacent loops with $60m$ and $90m$. We highlight specific places in each map that we use to describe the paths in a simplified way, as shown in Table 4.1. Note that these places do not correspond to the regions generated by our method, since, in our case, the regions correspond to segments of the environment where the density is homogeneous. In scenarios A and B, the robot has traveled for long periods of time, guided by the Potential Rails algorithm (MAFFEI et al., 2014)[5]. In scenario C, instead of a long run, our method tries to find matches among multiple small runs.

For all scenarios we performed experiments varying the size of the grams from 1 to 20. We also varied the thresholds for accepting matches of grams. Free-space densities

[5]Potential Rails is one of our works on integrated exploration using boundary value problem with focus on generating exploratory and patrolling behaviors for mobile robots (MAFFEI et al., 2014).

Figure 4.10: Scenarios used in the experiments.



(a) Scenario A (b) Scenario B

(c) Scenario C

Table 4.1: Robot paths in each scenario

Scenario A
ABCD ABCH GCDA BCHG BADC BABH GCDA BCHE DCGH DABC DEGB FEAB CHGB FHDA BCDE GBFE ABCH GBFH ABCD EGBF EABC HGBF HDAB CDEG CHGB FEAB CDAF HCGH DEGB ADCB FHDE GCDA BCHG BFED CBAD HEDC GHDA FGBA EHCB FDCG HDAB CDEG BFEA BCHG BFHD ABCD EGBF EABC HGBF HDAB CDEG BFEA B

Scenario B
AFGHIJ KLEBCD AFGHIJ KLEBCD AFGHIJ KLEBCD AFGHIJ KLEBCD AFGHIJ KLEBCD AFGHIJ KLEBCD AFGHLK JIEBCD AFGHLK JIEFGD ABCHEF KLIJGD ABCHEF GDABCH IJKLEF GDABCH LKJIEF GDABCH IJKLEF GDABCH IJKLIJ GDAFGH EBCDAF KLIJGH EBCDAF GHIJKL EBCDAF GHIJKL EBCDAF GHEFKL IJGDAB CHE

Scenario C	
(1)	DEBCEA DEBCEA DEBC
(2)	DEBCEA DEBCEA DEBCEA DEBC
(3)	DEBCEA DEBCEA DEBCEA DEBCEA D
(4)	DEADEB CEBCEA DEADE

were computed using a uniform kernel profile of radius $4m$. Three configurations of size tolerance for matches, ϵ_s, were tested: 10%, 20% and 30%.

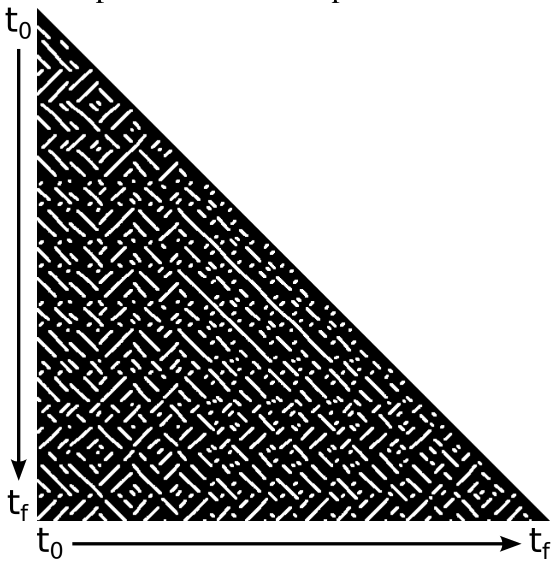
Before examining the results, we present in Figure 4.11 the similarity matrices associated to each scenario that indicate the moments of loop closure based on ground-truth. In each plot, the robot position at each step (vertical axis) is compared to all previous robot positions (horizontal axis). Matches are shown in white, while positions that do not match are shown in black (those correspond to the vast majority of situations). White lines descending to the right represent sequences of matches obtained in the same visiting order. In contrast, white lines descending to the left represent sequence of matches obtained in reverse direction. Our strategy only searches for matches in the same visiting order. Lastly, long white lines are associated to long sequences of matches, which our method can identify quite well, therefore by simply looking at Figure 4.11, we should expect the best results in Scenario C.

4.3.2 Analysis of Results

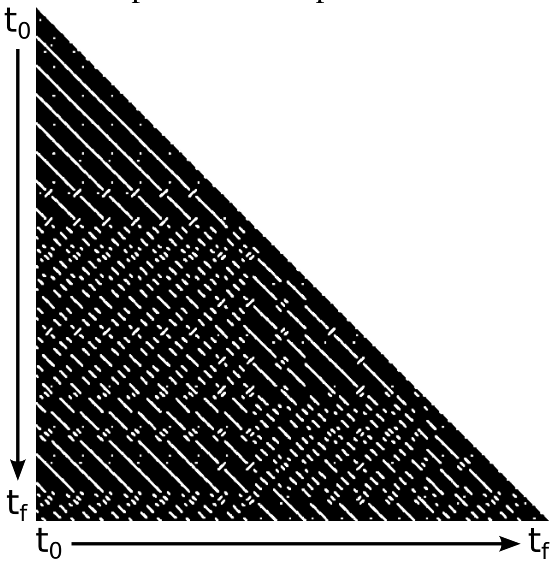
Results of precision and recall varying gram sizes and thresholds for scenario A are shown in Figure 4.12a. We can observe that the highest recall was obtained with the most tolerant threshold, i.e. 30%, but, as expected, with the smallest precision. In terms of gram size, the precision obtained using large sizes of grams was much higher than the precision obtained with small sizes. This happens because there is a reduction of ambiguities when the size of the matched sections of environment is increased. Figure 4.13a shows the variation of both precision and recall over time, considering different gram sizes and a threshold of 20%. We can see that using grams of large size the method may take a long time to detect matches, and, generally, the recall stays small. This can be problematic because sometimes using only few true matches in the graph optimization is not sufficient to obtain an adequate map. Still, it is better than situations with false matches. Lastly, we present the resulting topological map of Scenario A built using 12-grams. The ground truth, the raw odometry and the estimated map are respectively shown in Figures 4.14a, 4.14c and 4.14e. The obtained matches are topologically correct. However, there are regions with no matches, which cannot be properly corrected during the graph optimization. We could reduce the size of the grams to obtain more matches, although this measure increases the occurrences of false positives.

Figures 4.12b and 4.13b show the results obtained in Scenario B. In comparison with

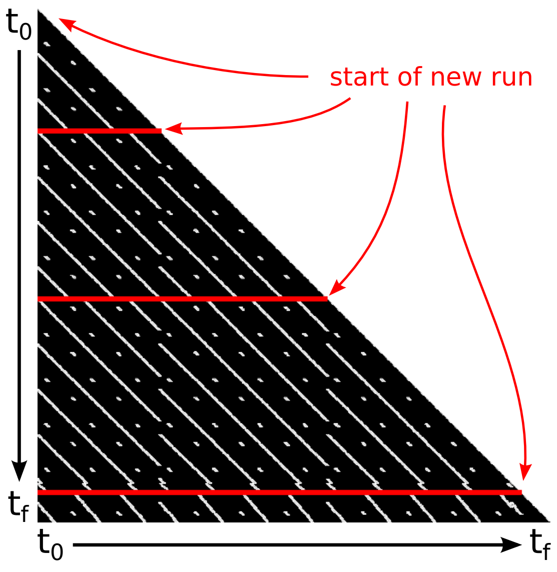
Figure 4.11: Similarity matrices associated to each scenario. The robot position at each instant (vertical axis) is compared to all previous robot positions (horizontal axis). White lines represent true correspondences. Scenario C is composed of multiple runs.



(a) Scenario A

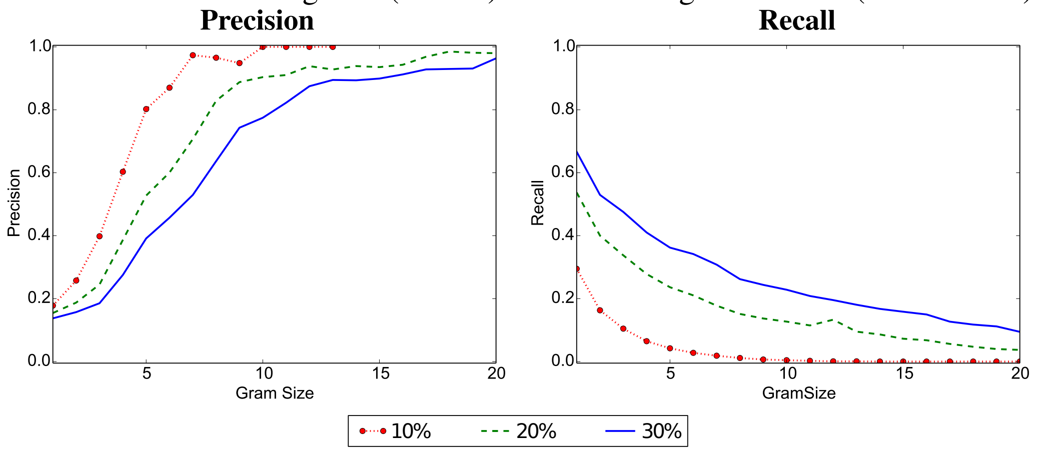


(b) Scenario B

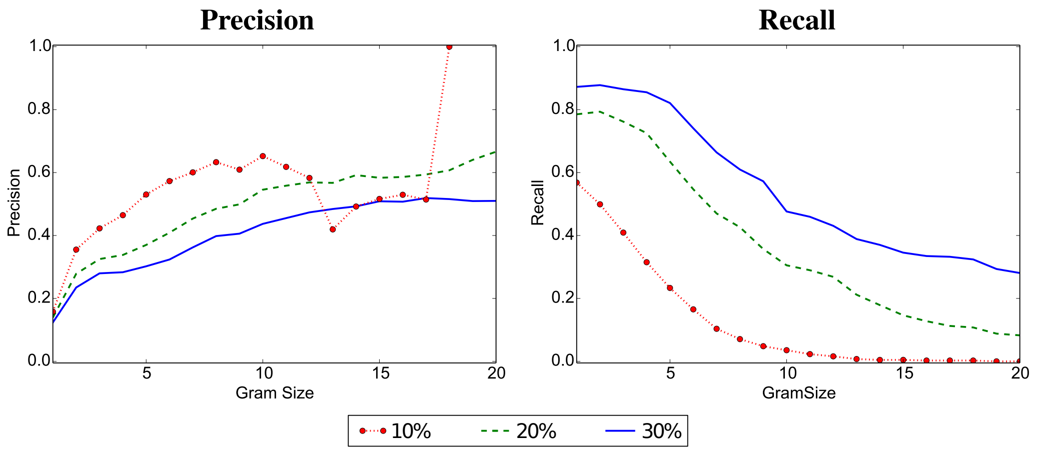


(c) Scenario C

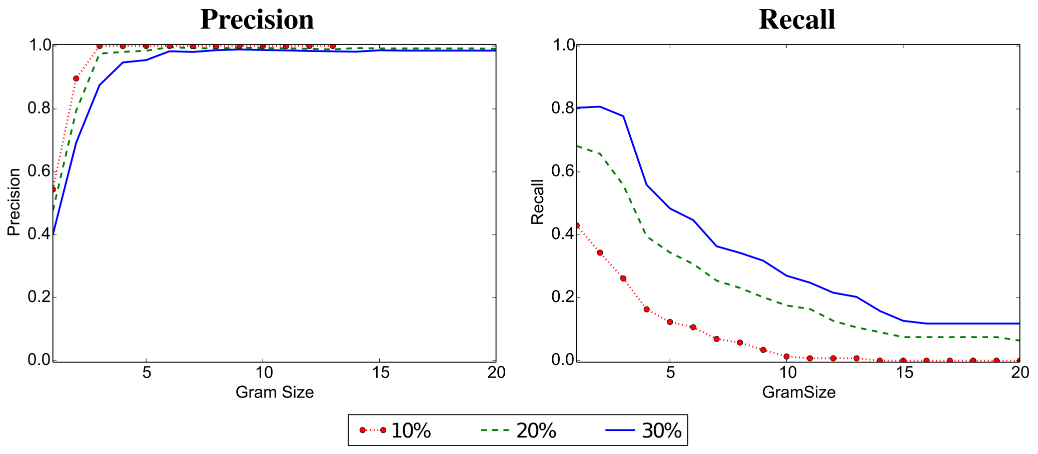
Figure 4.12: Results of *precision* and *recall* obtained in each scenario according to the variations in the size of the grams (1 to 20) and in the length threshold (10% to 30%).



(a) Scenario A



(b) Scenario B



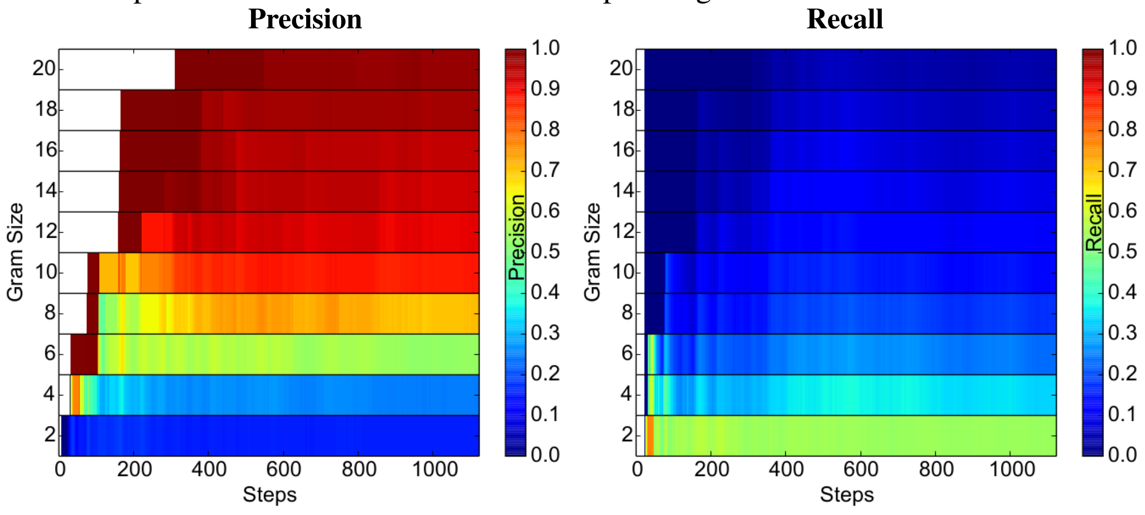
(c) Scenario C

Scenario A, the recall was better but the precision was much worse. Looking at Figure 4.13b, we can see that both precision and recall stay high until half of the path and decrease after such point. This can be explained if we look back into the trajectory described in Table 4.1. Its initial phase is highly repetitive, therefore it is easy for the method to obtain correct matches. The problem is that, just after the robot leaves its repetitive behavior, no match can be found, which instantaneously reduces the recall. The precision also drops due to erroneous matches related to the high level of symmetry in the map. Ideally, the larger the size of the gram, the higher the precision. Unfortunately, if we increase too much the minimum gram size to accept a match we tend to obtain zero recall, i.e. inability to find any match (either true or false). Examining the results obtained with the threshold of 10%, the precision decreases when using large grams. A possible explanation for this is that a false positive (expected to happen in symmetric maps) can drastically reduce the precision if the number of matches is too small, which is likely to occur using a small threshold and large size of grams. This environment represents the worst case scenario for our method, preventing the construction of a useful map.

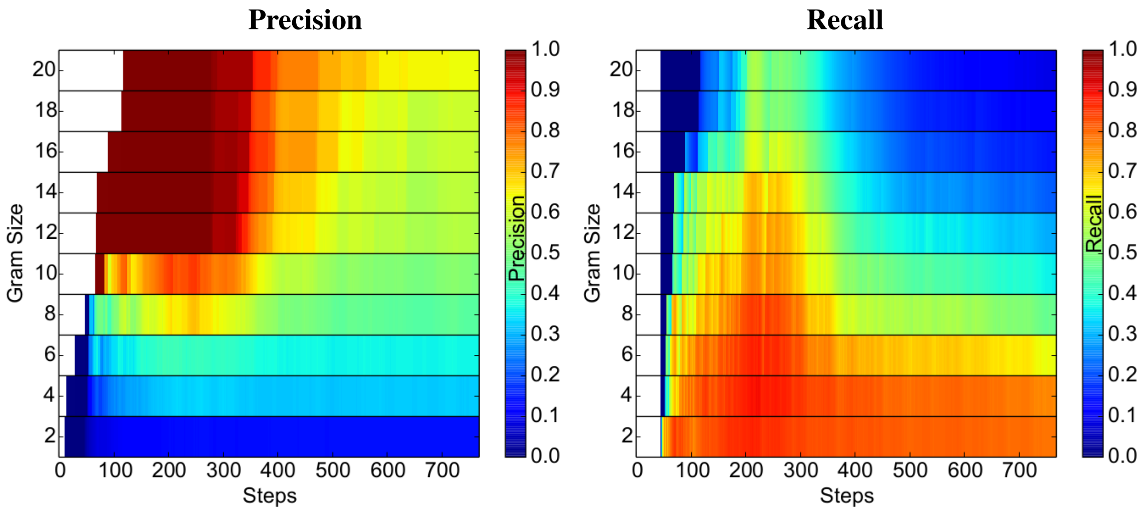
Finally, results of Scenario C are presented in Figures 4.12c and 4.13c. This dataset is the most consistent one, in spite of being obtained from multiple runs. That said, the precision is high even using a small gram size. The ground truth, the raw odometry and the resulting topological map of Scenario C built using 6-grams is shown in Figures 4.14b, 4.14d and 4.14f, respectively. The detected matches are correct, but they are too few, which prevents the construction of an accurate map. One cause for this problem is that the initial and final sections from each test cannot be matched, since they represent incomplete segments. Additionally, there is an important missing match in the intersection of the two loops, which cannot be solved by the method, because the robot visited this part of the map in different directions, i.e. generating different words. At this moment, this is a limitation of our method.

Nonetheless, Scenario C is specially interesting since it illustrates situations of multiple runs, in which the robot does not have any indication of its position in relation to the other trajectories. The only approach that can be taken in this case is to check all possibilities, which is impractical for traditional matching techniques such as ICP (RUSINKIEWICZ; LEVOY, 2001). In fact, we performed the ICP over points extracted from the same local map used in our approach, varying the acceptance tolerance, and the resulting precision was low. Table 4.2 shows a comparison between results obtained with our method and with ICP. The best result obtained by ICP in this scenario was a precision of 11% with a

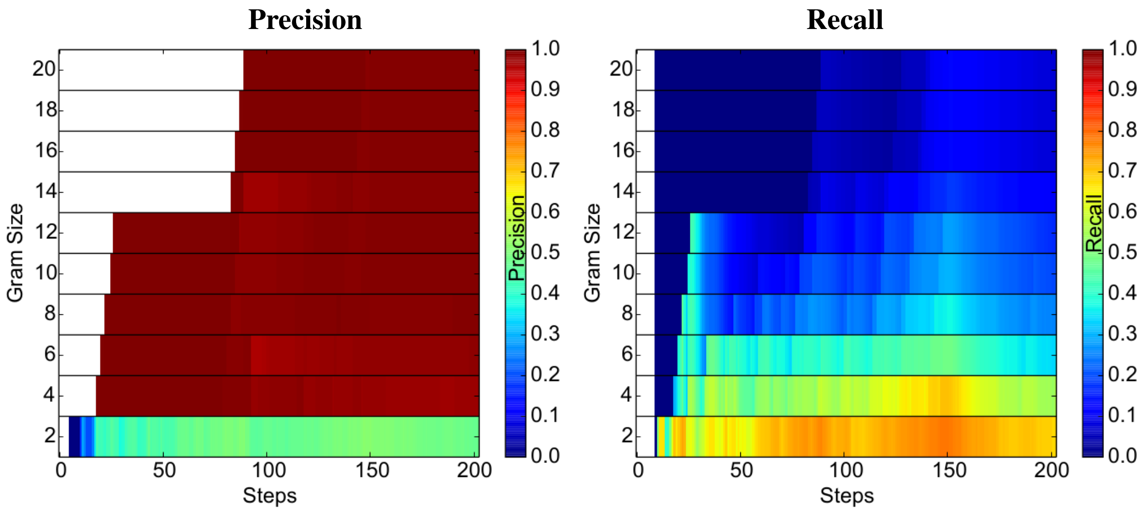
Figure 4.13: Variation of precision and recall over time in scenarios A, B and C. Each row corresponds to the values obtained with a specific gram size.



(a) Scenario A



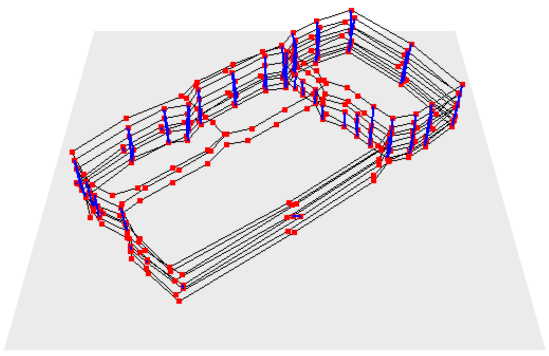
(b) Scenario B

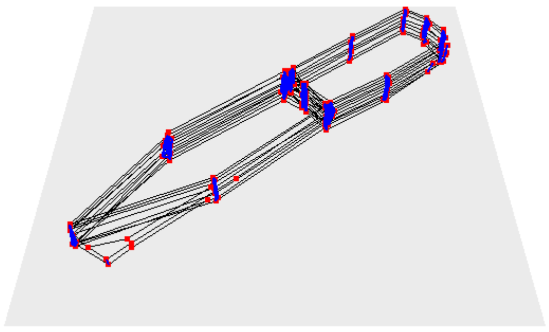


(c) Scenario C

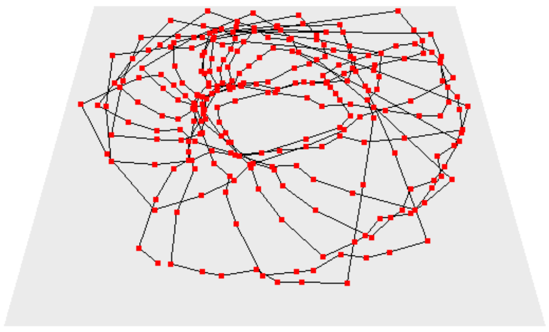
Figure 4.14: Topological maps of Scenarios A and C. Red nodes represent the median position of each word. Black lines represent edges connecting adjacent words. Blue lines represent edges connecting matched words.

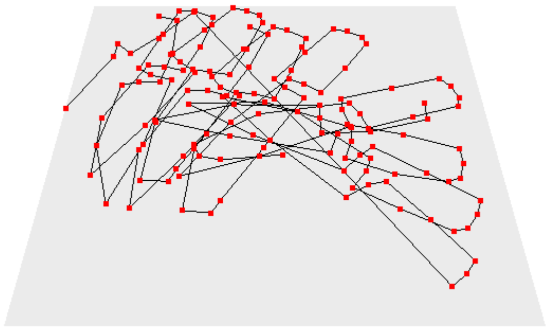
Ground Truth


(a) Scenario A

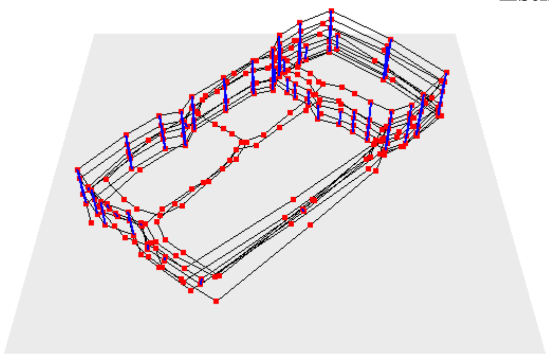

(b) Scenario C

Odometry


(c) Scenario A


(d) Scenario C

Estimated


(e) Scenario A

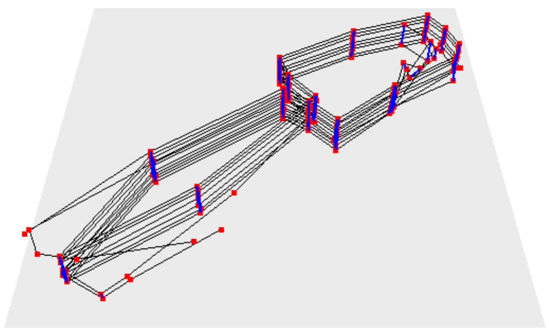

(f) Scenario C

Table 4.2: Comparison with ICP in Scenario C.

	Precision	Recall	Checkings per 100 msec
ICP 100 pts	0.1181	0.3568	30
20-gram	0.9822	0.3211	60.000.000

recall of 35%, while the best result obtained by our method was a precision of 98% with a recall of 32%. Another important point is that, once the words and the grams are built, the computation time required by our method is about six orders of magnitude lower than ICP, because the matching process is reduced to simple queries. In fact, the computation of free-space density and posterior creation of words are the slowest parts of the method – in average we compute the densities of 40 robot observations in an interval of 100 msec.

4.4 Related Work

Place recognition strategies are seen as the front-end of Simultaneous Localization and Mapping (SLAM) algorithms. An important element to be considered is an efficient observation model which is associated to the type of sensor been used. Another important element is the algorithm applied to match sets of observations into loop closures. The loop detection algorithm is concerned on how observations will be used to find loops. One way to approach the problem is to match the current observation with others to detect a loop (CUMMINS; NEWMAN, 2007; CUMMINS; NEWMAN, 2009). This, however, does not take into account the sequential nature of the problem, since revisiting a known location generally includes experiencing a similar sequence of observations. Evidences (HO; NEWMAN, 2007; MILFORD; WYETH, 2012) show that matching a sequence of observations can offer superior results for place recognition than unique matches alone. Even more, it has been shown that the analysis of sequential information can disambiguate seemingly indistinguishable regions (WERNER et al., 2009).

Vision-based approaches are extensively used for loop detection (CUMMINS; NEWMAN, 2009; MILFORD; WYETH, 2012; GALVEZ-LÓPEZ; TARDOS, 2012). FABMAP (CUMMINS; NEWMAN, 2007; CUMMINS; NEWMAN, 2009) combines feature detection and bag-of-words (BoW) to perform loop closing. Outdoor SLAM (HO; NEWMAN, 2007) matches sequences of similar frames and explore the time space correlations among them. SeqSLAM (MILFORD; WYETH, 2012) also performs sequence matching, but assumes that the robot moves with constant speed following the same path that was traversed before (e.g. like a train that moves over the same tracks). One of the key aspects behind SeqSLAM is the dimensionality reduction of sensor information (frame data), which is segmented into normalized intensity patches of a reduced image. Frame comparison is performed using the sum of absolute differences, which is robust to drastic illumination changes when the method's assumptions hold. Binary Visual BoW can be used to perform loop closing with high precision and online performance considering in-plane camera motion (GALVEZ-LÓPEZ; TARDOS, 2012). Again, the way sensor information is translated into an observation model is key to achieve performance improvements.

The use of range finders for place recognition is also possible. Even so, depending on how the observation model is computed and the size of the measurement vector, place recognition can still be computationally expensive. One way to match regions is to use the Iterative Closest Point (ICP) algorithm (NUCHTER et al., 2005) or one of its variations (RUSINKIEWICZ; LEVOY, 2001). However, matching a subset of points with the entire set of points is sometimes infeasible. Kumar et al. (KUMAR; GUIVANT; DURRANT-WHYTE, 2004) show that compact representation of the environment can be built through non-linear dimensionality reduction strategies, preserving perceptually meaningful structures of the environment. Chen et al. (CHEN; WANG, 2006) use Principal Component Analysis over every range finder scan, to convert scans into Gaussian distributions. They note that similar observations, encoded in low dimensional representations, tend to group themselves into distinct clusters having similar appearance. These clusters can be used to connect a topological map using Bayesian inference. Werner et al. (WERNER et al., 2009) use the transitive connection of nodes of the Voronoi diagram of the environment to disambiguate similar places. Nodes are labeled with symbols which represent sensor information (e.g. distance to nearest obstacle obtained with a sonar). Then, a particle filter based on n-grams (corresponding to sequences of nodes) is used to determine the smallest topological graph representing the environment.

Our approach is similar to Werner's approach (WERNER et al., 2009) by using n-grams to match sequences of regions in the environment. However, in our case, the words that compose the n-grams are defined by contiguous regions of similar free-space density. Another difference is that our method also encodes orientation and the traveled distance into each word. Finally, we extensively search for n-grams matches in an efficient way, given the highly reduced data that emerges from the construction of words. In next section, we present a review of the n-grams technique, originally from the field of speech and language processing, that plays a central role in our algorithm.

4.5 Summary

In this work the key contributions are:

(i) An efficient method for place recognition which matches sequences of words using a laser range finder;

(ii) A compact representation of sets of similar observations into a single word;

(iii) The word representation which includes topological neighborhood information, the number of observations and orientation;

(iv) A strategy to quantize FSD information, which results in segmentation of the environment into relevant topological structures;

Our experiments show that FSDs can be successfully used as main features to build words associated to regions in the map. The comparison of words is performed considering the number of observations, the angle variation, and the quantized FSD. The adoption of Mean Shift seems to segment the environment into relevant topological structures. Still, future studies should include strategies to deal with long term and abrupt changes in the probability distribution function of observations (histogram of densities). When Mean Shift quantization is performed, the topological graph must be reconstructed and minimized again from the start. This may result in no back-end convergence. Better strategies should be employed to take advantage of topological information acquired from previous quantizations. Additionally, we can choose to reconstruct the graph only when there is a significant change in the histogram of densities.

Furthermore, the strategy to determine the size of the n-gram to be used is still an open issue. Even though large sequences seem to present 100% precision, it is not advisable to increase the size of the n-gram indefinitely, since no matches might be found. On the other hand, considerably reducing the size of the n-gram seems to drastically impact the method's precision.

5 LONG-TERM PLACE RECOGNITION USING MULTI-LEVEL WORDS OF SPATIAL DENSITIES

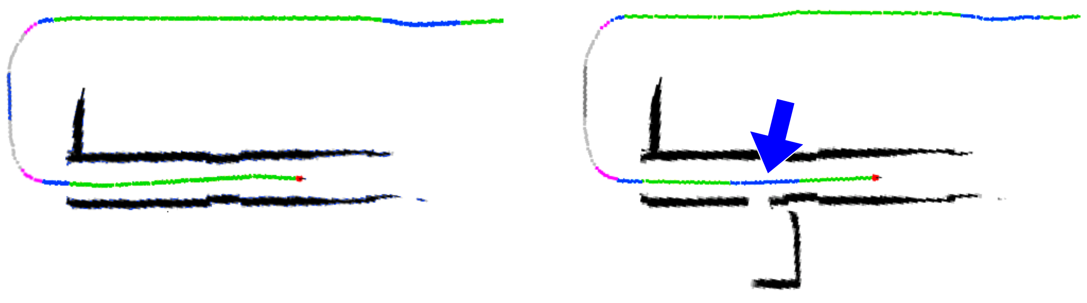
5.1 Introduction

Long-term mobile robot operation is a field that has attracted the interest of many roboticists (BIBER; DUCKETT, 2005; BIBER; DUCKETT, 2009; WALCOTT et al., 2012). With the increasing integration of robots in human activities, it is essential that they have the ability to adapt to changes in the environment. For example, a robot equipped with a laser range finder moving inside a building must differentiate static objects, such as walls and columns, from highly dynamic objects, such as people in motion. It must also differentiate both types of objects from semi-static objects, such as doors, tables, and other furniture, which can move occasionally.

Although most of the Simultaneous Localization and Mapping (SLAM) techniques work well in static environments and many techniques are able to deal with highly dynamic environments (e.g. by filtering moving objects in the robot's field-of-view (HAHNEL et al., 2003b)), the identification of semi-static objects is a very difficult problem. This comes from the fact that a robot cannot distinguish between static or dynamic objects that are standing still during a single visit to a place. Thus, revisiting an environment that suffered changes in the disposition of its objects will lead to conflicting sensor measurements, which can rather hinder the localization and mapping process. Nonetheless, such revisiting behavior is required in order to keep an updated map.

Some SLAM techniques that operate in dynamic environments divide the problem into building two distinct maps: one for static objects and other for dynamic objects (WOLF; SUKHATME, 2005; WANG et al., 2007). However, in long-term operation, objects can move in highly different time scales. Some authors propose a representation with multiple maps, where each one is updated in a specific time scale (BIBER; DUCKETT, 2005; BIBER; DUCKETT, 2009), others propose a single dynamic occupancy grid, which stores, for each grid cell, a probabilistic estimate of changes in the environment (TIPALDI et al., 2011; TIPALDI et al., 2012). A more recent approach is the dynamic pose graph (DPG) (WALCOTT et al., 2012), which is updated whenever changes in the environment occur, by inserting or removing nodes or connections between nodes. An important aspect of DPG is that it maintains a history of changes in the environment, instead of just keeping an updated current map.

Figure 5.1: Changes in the environment, such as an open/closed door, affect the word construction. The higher density region (in blue, pointed by the blue arrow) is only obtained when the door is open.



(a) Closed door (b) Open door

An assumption made by most long-term SLAM approaches is that the starting and ending point of the robot trajectory are known. In situations where this is not the case (e.g. when we simply start the robot navigation from an unknown place), the data association problem becomes much more difficult. A successful place recognition is fundamental for SLAM. In fact, the occurrence of a few false positives can fully degrade the solution (THRUN; BURGARD; FOX, 2005; GRISETTI et al., 2010b).

In the previous chapter, we introduce a place recognition method based on sequences of low dimensional observations acquired by a robot using a laser range finder. Such information is obtained by computing kernel density estimates (KDE) of the free space. The method quantizes the information into density classes, and generates simple words to represent regions of the environment. Place recognition is made by matching sequences of words. While the method obtains good results in structured environments and performs fast searches for long sequences of observations, it is focused on static environments.

When an environment is not static, we cannot assure that all information been observed will always be constant. Considering our idea of using words to represent regions, what happens if a word that was just created does not belong to a static object? Changes in the environment may affect the density estimates and degrade the performance of the word matching. For instance, when the robot is visiting the region in Fig. 5.1(a) and all doors in the corridor are closed, the method generates a single large *green* segment (described by a specific word) to represent such region. However, if there is an open door, as shown in Fig. 5.1(b), the method generates three words to represent the *green-blue-green* regions. Basically, we must consider the possibility of not always finding the same words when revisiting a region.

Our new approach[1] takes the idea of using words to represent regions and expands it for the problem of long-term operation. The proposal is that every time a word is built, we also try to build alternative words that would exist in the absence of the observed one. That is, when building a new word, we also consider the case where the density change was brought by a non-static object and create a new word sequence assuming the previous density remained present in the new word's location. Place recognition is made by searching n-grams of words (both real or alternatives). Experiments performed in real and simulated scenarios are shown, and demonstrate the advantages associated to the use of multi-level words.

We begin this chapter detailing the expansion for long-term place recognition of our previous method based on n-grams, followed by a density-based strategy used for fine tuning of matches. Next, we present the results of experiments in long-term operation. Finally, we discuss related work and draw our conclusions.

5.2 Building multi-level words of spatial densities for place recognition in lifelong operation

When the robot is in lifelong operation, changes in the environment affect the density estimates. In this case, words that describe the same place but are generated at different occasions will not match. We circumvent this problem in a multi-level approach by generating alternative (and possibly corrected) words for all regions. Basically, we start with the case where one word is wrong, such as the open door in Figure 5.1(b), and expand it to cases where multiple consecutive words are wrong.

5.2.1 Extending the previous definition of word

Before proceeding with the details of our strategy, we need to introduce a new definition of word, $\mathbb{W}$, extending the previous definition of word, $\mathtt{W}$, presented in Equation 4.3 of Section 4.2.2.

$$\mathbb{W} = \langle \mathtt{d}, \mathtt{s}, \mathtt{a}, \mathtt{P}, \mathtt{O} \rangle \tag{5.1}$$

[1]The core idea of this chapter was originally published in the paper "Long-term place recognition using multi-level words of spatial densities" (MAFFEI et al., 2016), presented at the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'16).

The new word is composed of the three original syllables of W (i.e. the density class d, the size s, and the angle variation a) along with two more elements:

- P: the list of predecessor words, that is, the real and alternative words immediately before the word W. In the single-level approach presented last chapter, each word has only one predecessor, which can be obtained directly (i.e. the predecessor of the word $\mathtt{W}_n$ is the word $\mathtt{W}_{n-1}$). In the novel multi-level approach, besides the previous real word that is always a predecessor, each word can have multiple alternative words as predecessors (at most one for each level), thus we need to explicit state what those words are.

- O: the list of observations that compose the region described by the word W. In the single-level approach, we do not need to maintain such information after the creation of the word. However, in the multi-level approach, the list of observations associated to existing words may be used in the future to create alternative words.

We also redefine the algorithm for building a word given a contiguous region R associated to observations from same density class. The new Algorithm 5.1 is almost the same as the original Algorithm 4.3. The syllable d corresponds to the density class of the initial observation in R (line 1). The syllable s is the size of the region in terms of number of observations (line 2). The syllable a is the difference of the angle between median and final poses of the sequence of observations, and the angle between initial and median poses (line 3). The list of predecessors words, P, is initially empty (line 4). Posteriorly, P will be updated in Algorithm 5.2 for building multi-level words. Lastly, the list of observations, O, is associated to the observations in R (line 5).

Algorithm 5.1: Build Extended Word

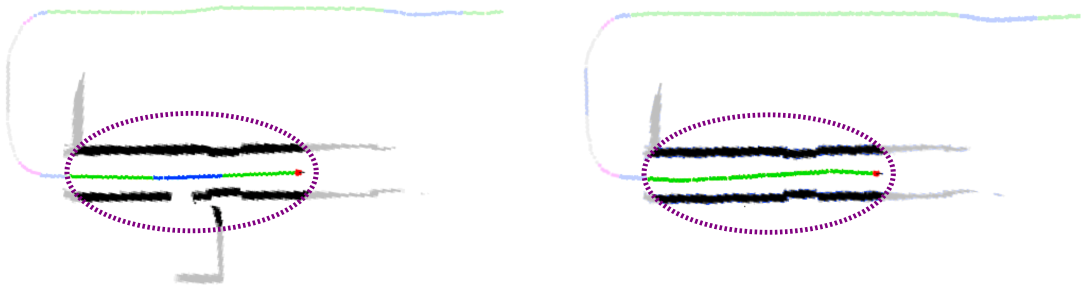
```
   Input: $R = [\boldsymbol{o}_i, \cdots, \boldsymbol{o}_{m=(i+f)/2}, \cdots, \boldsymbol{o}_f]$
   Output: $\mathbb{W}$
1  $\mathtt{d} = d(\boldsymbol{o}_i)$
2  $\mathtt{s} = size(R)$
3  $\mathtt{a} = arctan\left(\frac{y(\boldsymbol{x}_f)-y(\boldsymbol{x}_m)}{x(\boldsymbol{x}_f)-x(\boldsymbol{x}_m)}\right) - arctan\left(\frac{y(\boldsymbol{x}_m)-y(\boldsymbol{x}_i)}{x(\boldsymbol{x}_m)-x(\boldsymbol{x}_i)}\right)$
4  $\mathtt{P} = [\ ]$
5  $\mathtt{O} = R$
6  $\mathbb{W} = \langle \mathtt{d}, \mathtt{s}, \mathtt{a}, \mathtt{P}, \mathtt{O} \rangle$
7  return $\mathbb{W}$
```

5.2.2 Building multi-level words

Our strategy for long-term place recognition generates words in different levels. In the l-th level, we detect sequences of $l+2$ words (i.e. $l+2$-grams) which start and end with words of the same density class but have l words in between them with different density classes. Those sequences are then fused into a larger word having the same density class. That is, we generate alternative words replacing l consecutive words. For instance, when the robot is visiting the region in Figure 5.2(a), the method will generate three words for the *green-blue-green* segment of the trajectory, but it will also generate a larger single *green* word and associate it to the same segment. This is a guess of a possible word that could occur due to some variation in the environment, such as a closed door, as shown in Figure 5.2(b). The description of the words are presented in Figures 5.2(c) and (d), and later in this section they will be further analyzed.

Algorithm 5.2 presents the strategy for building multi-level words, which runs always that a new contiguous region R is built. The algorithm's input is the new contiguous region R, along with the current lists of words in each level, $\mathcal{W}_0, \mathcal{W}_1, \cdots \mathcal{W}_L$, where $\mathcal{W}_l = [\mathbb{W}_{0,l}, \mathbb{W}_{1,l}, \cdots \mathbb{W}_{n,l}]$ with $\mathbb{W}_{j,l}$ being the j-th word of the l-th level. The first step is the creation of a new *level-0* word, $\mathbb{W}_{new}$, i.e. the real word (line 1). Then, we must add the last *level-0* word, $\mathbb{W}_{last}$, to the list of predecessors of $\mathbb{W}_{new}$ (lines 2-3) and add $\mathbb{W}_{new}$

Figure 5.2: Example of a region described by three words in a given situation, (a), and by an alternative word in another potential situation, (b). At the bottom are presented the descriptions of the three original words at level 0, (c), and of the alternative word at level 1, (d).



(a) (b)

$$\begin{aligned}\mathbb{W}_{n-2,0} &= \langle A,\ 12, -2,\ \mathtt{P}_A, R_{n-2}\rangle\\ \mathbb{W}_{n-1,0} &= \langle B,\ 9,\ \ 2,\ \mathtt{P}_B, R_{n-1}\rangle\\ \mathbb{W}_{n,0} &= \langle A,\ 10,\ \ 1,\ \mathtt{P}_C, R_n\rangle\end{aligned}$$

(c)

$$\mathbb{W}_{m,1} = \langle A, 31, 0.5, \mathtt{P}_A, (R_{n-2} \oplus R_{n-1} \oplus R_n)\rangle$$

(d)

to the list of *level-0* words, $\mathcal{W}_0$ (line 5).

Algorithm 5.2: Build Multi-level Words

```
   Input: $\mathcal{W}_0, \mathcal{W}_1, ..., \mathcal{W}_L, R$
   Output: $\mathcal{W}_0, \mathcal{W}_1, ..., \mathcal{W}_L$
1  $\mathbb{W}_{new} = BuildExtendedWord(R)$
2  $n = size(\mathcal{W}_0)$
3  $\mathbb{W}_{last} = \mathbb{W}_{(n-1),0}$
4  $push\_back(\mathbb{W}_{last}, \mathtt{P}(\mathbb{W}_{new}))$
5  $push\_back(\mathbb{W}_{new}, \mathcal{W}_0)$
6  for $l$ in $1...L$ do
7      $\mathbb{W}_{first} = \mathbb{W}_{(n-l-2),0}$
8      if $\mathtt{d}(\mathbb{W}_{first}) == \mathtt{d}(\mathbb{W}_{last})$ then
9          $R' = \mathtt{O}(\mathbb{W}_{first}) \oplus \cdots \oplus \mathtt{O}(\mathbb{W}_{last})$
10         $\mathbb{W}_{alt} = BuildExtendedWord(R')$
11         $\mathtt{P}(\mathbb{W}_{alt}) = \mathtt{P}(\mathbb{W}_{first})$
12         $push\_back(\mathbb{W}_{alt}, \mathtt{P}(\mathbb{W}_{new}))$
13         $push\_back(\mathbb{W}_{alt}, \mathcal{W}_l)$
14 return $\mathcal{W}_0, \mathcal{W}_1, ..., \mathcal{W}_L$
```

Next, for each l upper level, we check if the last *level-0* word ($\mathbb{W}_{last}$) and the *level-0* word that is $l+1$ positions before it ($\mathbb{W}_{first} = \mathbb{W}_{(n-l-2),0}$) have the same density class (lines 7-8). If this is the case, the new alternative word, $\mathbb{W}_{alt}$, is computed using the concatenation[2] of all observations associated to the last $l+2$ words (lines 9-10). After, the list of predecessors for $\mathbb{W}_{alt}$ is copied from the word at *level-0* having the same start position ($\mathbb{W}_{first}$), given that they share the same predecessors (line 11). Additionally, the new alternative word is added to the predecessors list of the recently created *level-0* word (line 12). Lastly, $\mathbb{W}_{alt}$ is added to the list of *level-l* words, $\mathcal{W}_l$ (line 13).

We look back at the example in Figure 5.2 to further clarify the multi-level word construction process. As described in Figure 5.2(d), the alternative *level-1* word has the same density class ($\mathtt{d} = A$) of the first and last *level-0* words, described in Figure 5.2(c). The size of the alternative word ($\mathtt{s} = 31$) is the sum of the sizes of the merged *level-0* words. This is expected given that the observations associated to the alternative word is the concatenation of all observations associated to the last three *level-0* words ($\mathtt{O} = R_{n-2} \oplus R_{n-1} \oplus R_n$). The angle variation is near zero in the example ($\mathtt{a} = 0.5$), but note that this is something totally dependent on the shape of the robot trajectory. The remaining element, which is the list of predecessors ($\mathtt{P} = \mathtt{P}_A$), is equal to the one associated

[2]As previously defined in Section 4.2.1, we use the symbol $\oplus$ as the operator to concatenate lists.

to the first *level-0* word.

5.2.3 Performing place recognition

Place recognition is performed by searching n-grams combining multi-level words. We define an n-gram match,

$$\mathcal{g} = [\langle \mathbb{W}_i, \mathbb{W}_u \rangle, \langle \mathbb{W}_{i+1}, \mathbb{W}_{u+1} \rangle, \cdots, \langle \mathbb{W}_{i+n-1}, \mathbb{W}_{u+n-1} \rangle], \tag{5.2}$$

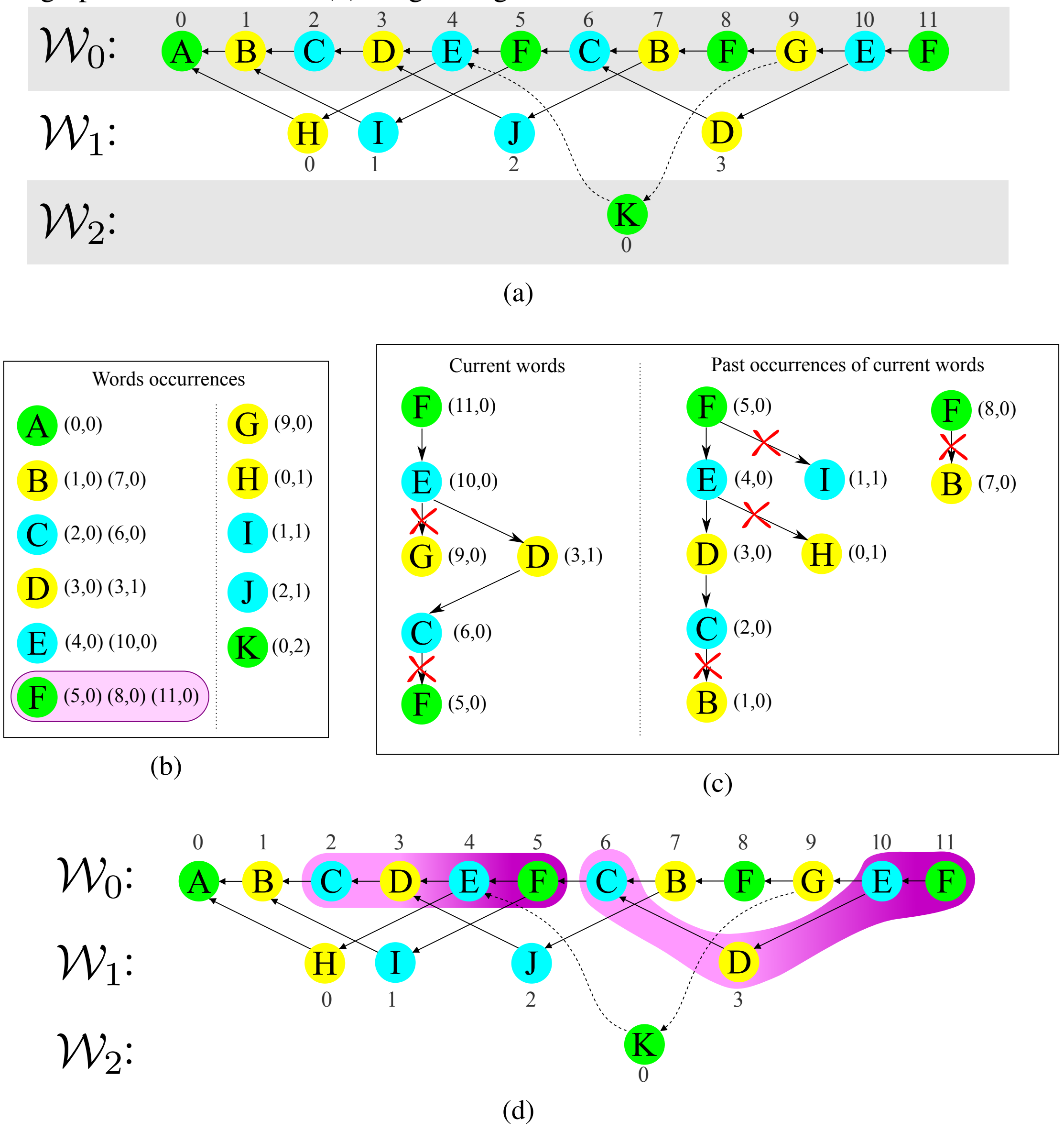
as a list of sequential pairs of words that can be truly matched according to the rules used in our previous approach, that is, with specific tolerances for each one of the three syllables $(\mathtt{d}, \mathtt{s}, \mathtt{a})$, as presented in Section 4.2.3. We also define $\mathcal{M}$ as the list of all matches of n-grams that occur at each instant. Like in our previous approach, we set a minimum number of words, $\epsilon_{\mathcal{g}}$, that must be reached to accept a match. Small n-gram sizes lead to large number of matches, but potentially low precision, while large ones give rise to low recall. Thus, determining good thresholds is an important aspect to take into consideration.

An important aspect of the novel strategy is that we tend to generate more matches by using more levels of words because we expand the number of possible n-grams that can be created. Therefore, n-grams composed of alternative words usually are more easy to match than n-grams (of the same size) composed of words from the lowest level. This is good for improving the recall, but prejudicial to the precision. Therefore, we must use different thresholds for accepting matches of n-grams composed of words from different levels. Due to this fact, the search for matches is incremental regarding the levels of words, i.e., at first it is performed considering only the lowest level, then it is performed considering the two lowest levels, and so on. We do this to keep matches obtained in the lowest levels that could be lost when checking matches in higher levels.

Algorithm 5.3 presents the strategy for matching multi-level words, while Figure 5.3 shows an example of the matching process with 12 *level-0* words, 4 *level-1* words and 1 *level-2* word. In the example, there are 11 different types of words (represented from A to K) that are divided in three different density classes (e.g. green, blue or yellow nodes).

The search starts by checking for matches of the last *level-0* word, $\mathbb{W}_{last}$, which in Figure 5.3(a) is $\mathbb{W}_{11,0}$ (of the type F). Instead of sweeping the whole trajectory, we speed up the search by keeping a list of lists of word occurrences, $\mathcal{O}$, which is exemplified in

Figure 5.3: Example of n-grams matching using multi-level words. (a) Search of matches start from the last word from *level-0*. (b) Occurrences of each word. (c) Search of matches through predecessors words (d) Largest n-gram match.



Algorithm 5.3: Find matches of multi-level words n-grams

```
   Input: $\mathbb{W}_{last}, \mathcal{O}$
   Output: $\mathcal{M}$

1  $\mathcal{M} = [\ ]$
2  $\mathcal{M}_{temp} = [\ ]$

3  for $\mathbb{W}_{matched}$ in $occurrences(\mathcal{O}, \mathbb{W}_{last})$ do
4      $\boldsymbol{g} = [\langle \mathbb{W}_{last}, \mathbb{W}_{matched} \rangle]$
5      $push\_front(\boldsymbol{g}, \mathcal{M}_{temp})$

6  while $\mathcal{M}_{temp} \neq [\ ]$ do
7      $\boldsymbol{g} = pop\_front(\mathcal{M}_{temp})$
8      $s = size(\mathcal{M}_{temp})$
9      $\langle \mathbb{W}_I, \mathbb{W}_{II} \rangle = front(\boldsymbol{g})$

10     for $\mathbb{W}_{prevI}$ in $\mathtt{P}(\mathbb{W}_I)$ do
11         for $\mathbb{W}_{prevII}$ in $\mathtt{P}(\mathbb{W}_{II})$ do
12             if $\mathbb{W}_{prevI} == \mathbb{W}_{prevII}$ then
13                 $push\_front(\langle \mathbb{W}_{prevI}, \mathbb{W}_{prevII} \rangle, \boldsymbol{g})$
14                 $push\_front(\boldsymbol{g}, \mathcal{M}_{temp})$

15     if $size(\mathcal{M}_{temp}) == s \wedge size(\boldsymbol{g}) > \epsilon_{\boldsymbol{g}}$ then
16         $push\_front(\boldsymbol{g}, \mathcal{M})$

17 return $\mathcal{M}$
```

Figure 5.3(b). For each occurrence of words that are equal to the word $\mathbb{W}_{last}$, we generate a new match $\boldsymbol{g}$ that is added to a list, $\mathcal{M}_{temp}$, of temporary matches to be posteriorly expanded (lines 3-5). In the example of Figure 5.3, the 5-th and 8-th words of the *level-0* are equal to $\mathbb{W}_{11,0}$, thus our method starts with the matches $\langle \mathbb{W}_{11,0}, \mathbb{W}_{5,0} \rangle$ and $\langle \mathbb{W}_{11,0}, \mathbb{W}_{8,0} \rangle$.

Following this one-word match, we just extend the search visiting the predecessors of the matched words. From each match $\boldsymbol{g}$ in $\mathcal{M}_{temp}$, we get the front pair of words, $\langle \mathbb{W}_I, \mathbb{W}_{II} \rangle$, (lines 7-9), and check if there is some pair of equal predecessors, $\langle \mathbb{W}_{prevI}, \mathbb{W}_{prevII} \rangle$, (lines 10-12). If this is the case, we add each pair of equal predecessors to $\boldsymbol{g}$ (line 13). Back in our example, one of the initial matched pair of words ($\langle \mathbb{W}_{11,0}, \mathbb{W}_{5,0} \rangle$) has a pair of equal predecessors ($\langle \mathbb{W}_{10,0}, \mathbb{W}_{4,0} \rangle$), therefore we can expand $\boldsymbol{g}$ to a bigram.

Our search continues recursively through the predecessors of $\mathbb{W}_{prevI}$ and $\mathbb{W}_{prevII}$ by adding the expanded n-gram match to $\mathcal{M}_{temp}$ (line 14). When it is not possible to expand $\boldsymbol{g}$, i.e. there are no pairs of equal predecessors, and the size of $\boldsymbol{g}$ is larger than a given threshold $\epsilon_{\boldsymbol{g}}$, we add $\boldsymbol{g}$ to the final solution $\mathcal{M}$ (lines 15-16). In our example, the largest

n-gram match is a sequence of four words (of the types $C \leftarrow D \leftarrow E \leftarrow F$),

$$[\langle \mathbb{W}_{6,0}, \mathbb{W}_{2,0} \rangle, \langle \mathbb{W}_{3,1}, \mathbb{W}_{3,0} \rangle, \langle \mathbb{W}_{10,0}, \mathbb{W}_{4,0} \rangle, \langle \mathbb{W}_{11,0}, \mathbb{W}_{5,0} \rangle].$$

Note that the third match (reading from right to left) happens between a word from *level-0* and an alternative word from *level-1*. If we only considered matches of *level-0* words (as in our previous single-level approach), the solution would be interrupted in the second match of words.

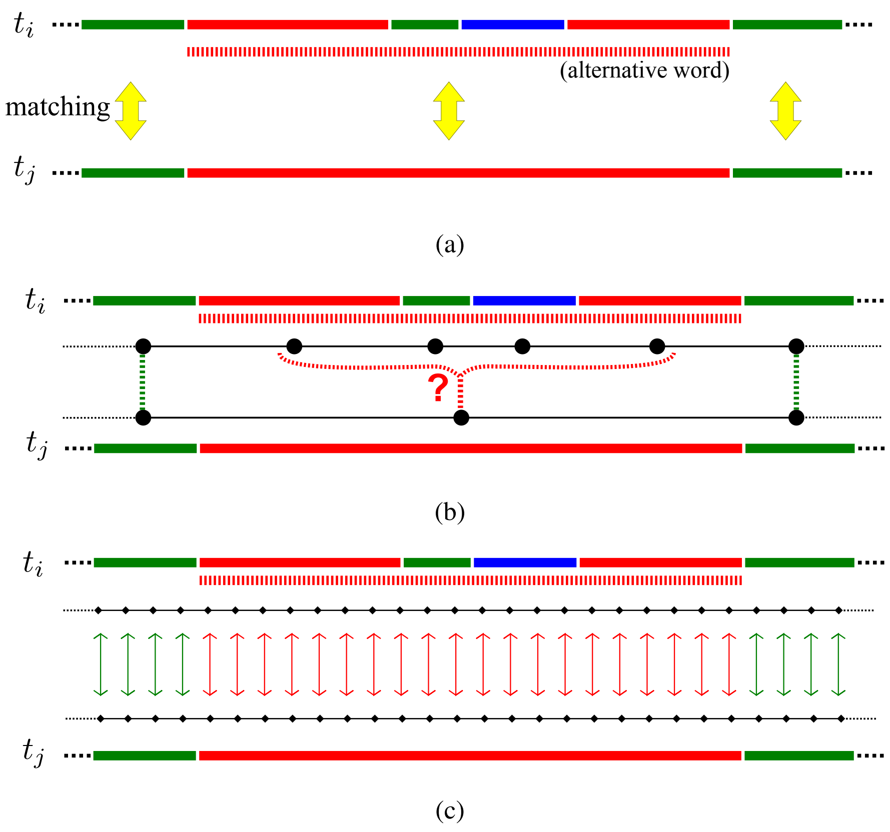
5.2.4 Fast adjustment of matches by evaluating raw spatial densities

Our strategy of density-based words to represent regions of an environment focus on finding topologically correct matches. As shown last chapter, the original approach associates each region (described by a single word) with one node in the pose graph representing the robot trajectory. Then, word matches simply connect the nodes associated to the words. With our new approach, we sometimes describe regions using alternative words that may encompass multiple original words.

A possible graph-building strategy could be to create nodes for the original words (exactly like the previous approach), and to find proper correspondences between nodes associated to original and alternative words. However, as exemplified in Figure 5.4, this strategy can be problematic. Figure 5.4(a) shows a match of a 3-gram composed of original words (obtained at time t_j) with a 3-gram that includes an alternative word (obtained at time t_i). The alternative word from time t_i is equivalent to four original words, thus, following the proposed strategy, we have the problem of trying to associate one node from time t_j with four nodes from time t_i, as shown in Figure 5.4(b).

We could avoid this problem by creating one node for each word, disregarding if it is original or alternative. This strategy increases the number of nodes in the graph in proportion to the number of levels of alternative words, which is not bad when only using a few levels. Nevertheless, a major problem of this strategy is that we may insert too much error in the pose graph by associating single nodes with alternative words that represent very long regions. Note that, during the graph construction, the pose associated to each node comes from odometry readings, therefore, the error between very distant adjacent nodes may be too big to be corrected during the graph optimization. Also, another point of adopting only a topological view is that the match of any two regions have the

Figure 5.4: Association of regions and nodes from pose-graph. (a) Example of match of a 3-gram including a region described by an alternative word. (b) Problem that arise when each original word is associated to one node. (c) Association of each region to uniformly spaced nodes.

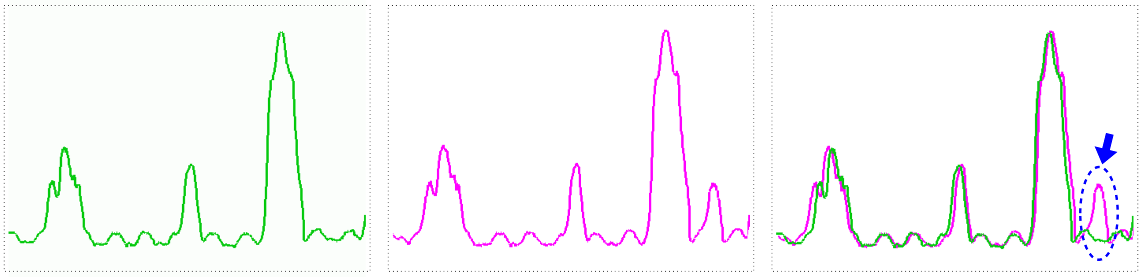


same importance. However, the impact of matching different sizes of regions drastically influences the quality of a final metric map.

In short, while finding correct topological correspondences is a fundamental task, this time we also deepen the investigation of translating our n-grams matches into more precise metric matches. Our idea is to translate all regions to equivalent sets of points, which are generated in constant intervals. When a match occurs between two words, the regions represented by those words are converted into subsets of the existing set of points that must be connected. Figure 5.4(c) illustrates the association of regions and nodes for the example of 3-grams matching. As we can see, it does not matter if words are original or alternative, with this approach the main problem consists in determining a proper alignment between two node sequences that have a size proportional to the size of the matched regions.

In order to find a proper connection between node sequences, we propose a fine tuning method using raw densities values. When we quantize the raw densities signatures to build

Figure 5.5: Variation of raw density values (Ψ) obtained in the regions shown in Figure 5.1. In (c), all values match well, with exception of the small bump associated to the open door.



(a) closed door (b) open door (c) alignment

words, we are simplifying the information to facilitate the matching of large trajectory segments. However, we can still compare the raw density values of the observations associated to the words to obtain better similarity measures. Therefore, we search the best alignment, δ, between two sequences of observations that minimizes the sum of the squared error between density values,

$$\delta = \underset{-\delta_{Max}<d<\delta_{Max}}{\operatorname{argmin}} \sum_{i=0}^{n} \left(\Psi(\boldsymbol{o}_{(s1+i+d)}) - \Psi(\boldsymbol{o}_{(s2+i)})\right)^2, \tag{5.3}$$

where $s1$ and $s2$ are the starting observations of both words, n is the size of the smallest word between the two, and δ_{Max} is the maximum displacement that we are searching.

Figure 5.5 shows an example of raw density values from the matched regions presented in Figure 5.1. As we can observe, the two curves of density values are similar, with the only significant difference occurring in the exact position associated to the open door (as pointed by the blue arrow). Still, considering that changes in the environment are generally small and somewhat distant from each other, such alignment method can obtain good results.

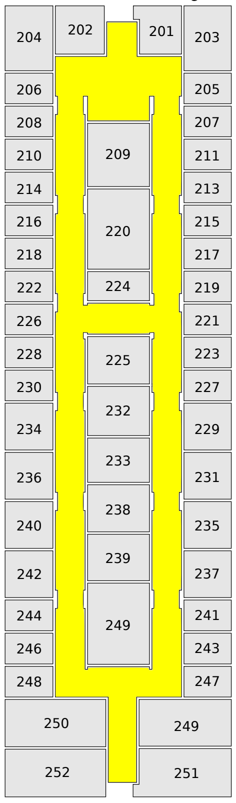
5.3 Experiments

5.3.1 Evaluation Scenarios

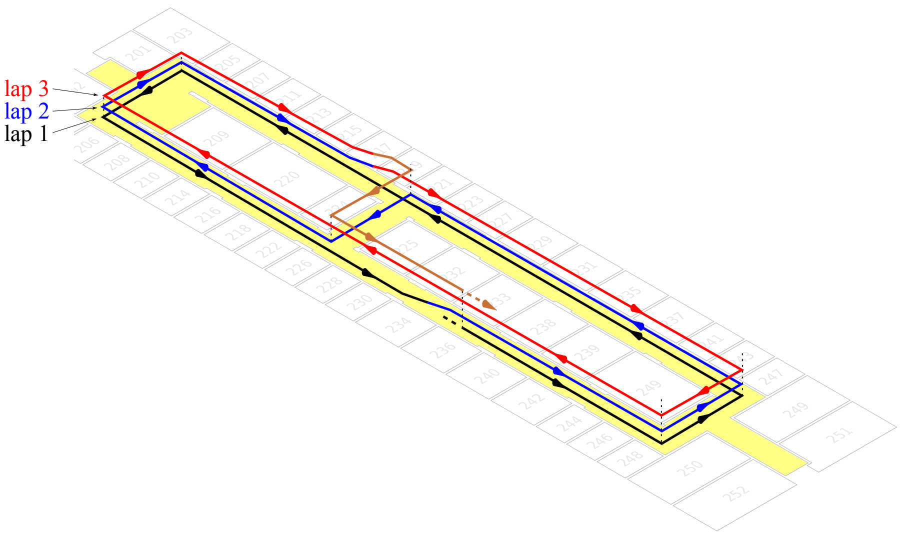
We evaluated our method using a Pioneer 3-DX mobile robot equipped with a SICK LMS 200 laser range finder and a notebook with a Intel® QuadCore™ i7 processor with 16GB of RAM memory, as described in AppendixA. All tests were made considering the same configuration for generating words: kernel radius of $2.5m$ for estimating the free-space density, density quantization into 8 classes, and size and angle thresholds during matches of 25% and $30°$, respectively.

The robot performed multiple runs starting from different positions in the environment shown in Figure 5.6(a), which is an indoor environment in our university. Regarding the navigation used in the tests, the robot always went forward in corridors and corners, and turned in 50% of the times that it has reached a bifurcation (there are only two bifurcations in the middle of the environment, near rooms 226 and 221).

Figure 5.6: Environment used in the experiments, (a), along with an example of three laps in the robot trajectory, (b).



(a)

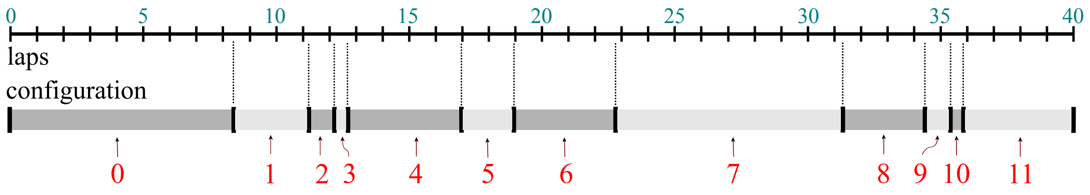


(b)

Three scenarios were chosen to evaluate the performance of the method. *Scenario A* is a real unsupervised scenario, where the measurement data was obtained without any control on the dynamics of the environment, such as open/closed doors, or people passing by the robot. *Scenarios B* and *C* are simulated scenarios that suffer changes in the environment from time to time. The robot performed around 8 laps in *Scenario A*, and 40 laps in *Scenarios B* and *C*, summing up several runs starting at different points in the environment. We use the term *lap* to describe the full external circuit of the environment, that is, a circuit passing one time by each of the four long corridors of the environment.

For instance, Figure 5.6(b) shows an example of three laps performed by the robot.

Figure 5.7: Variation of environment configurations in *Scenarios B* and *C*. (a) There are 12 changes of environment during the 40 laps covered by the robot in each test. (b) Description of the configurations used in each scenario.



(a)

Configuration	0	1	2	3	4	5	6	7	8	9	10	11
Length (%)	21.4	7.1	2.4	1.2	10.7	4.8	9.6	21.4	7.1	2.4	1.2	10.7
Scenario B	E_0	E_{B1}	E_{B2}	E_0	E_{B1}	E_{B2}	E_{B1}	E_0	E_{B1}	E_0	E_{B2}	E_{B1}
Scenario C	E_0	E_{C1}	E_{C2}	E_{C3}	E_{C4}	E_{C5}	E_0	E_0	E_{C1}	E_{C2}	E_{C3}	E_{C4}

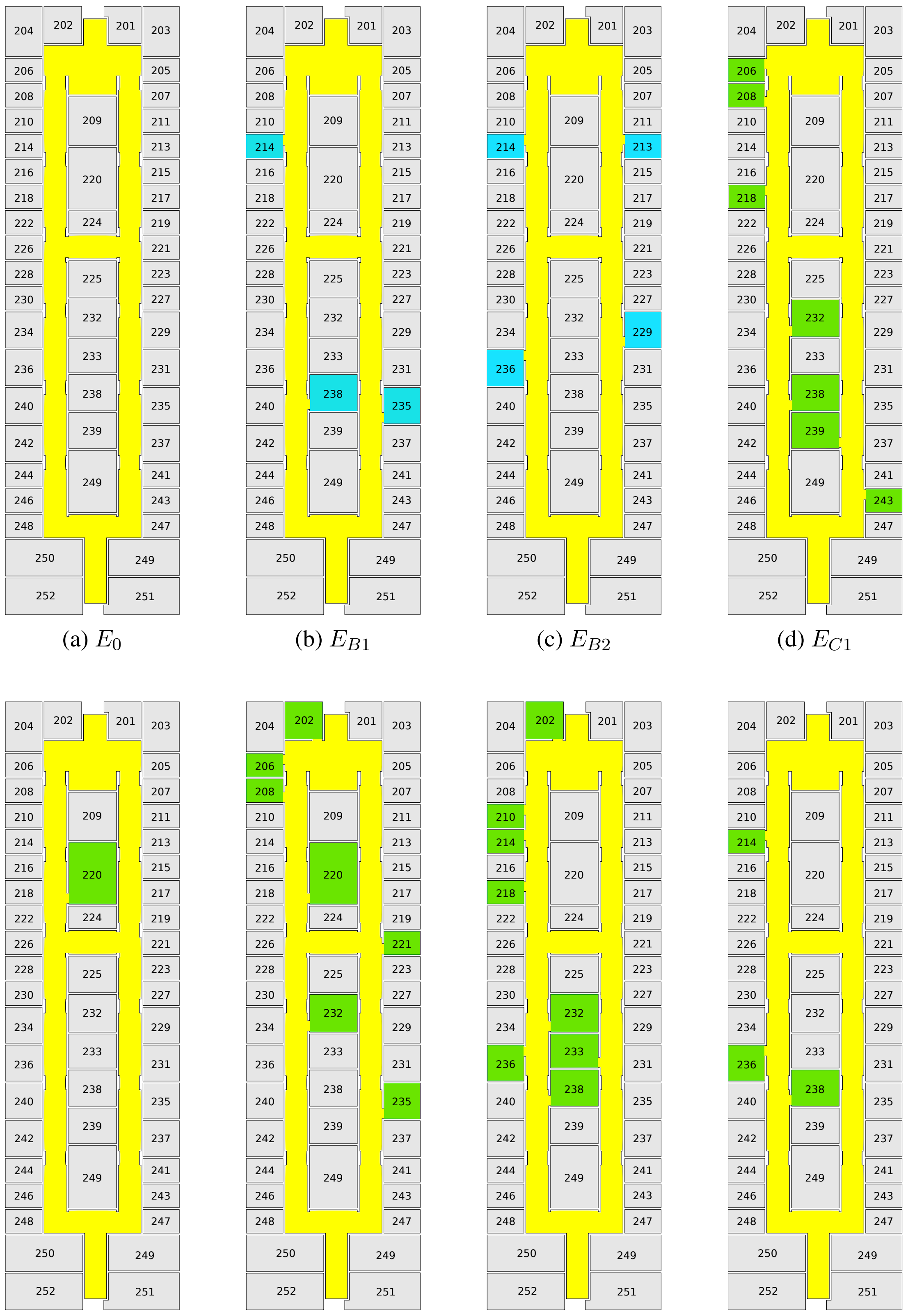
(b)

Further discussing the simulated scenarios, we defined some environment configurations (e.g. E_{B1}, E_{B2}, E_{C1}, E_{C1}, ...) representing variations of semi-static objects, in this case, rooms with open doors, as illustrated in Figure 5.8. During the tests in each scenario, 12 configuration changes occurred in the period of 40 laps. The configurations also occurred for different time lengths, i.e., some configurations were keeped for long periods of time, while others were momentary. As shown in Figure 5.7, the initial configuration stays for 21.4% of the test (from lap 0 to 8), the next configuration stays for 7.1% (from lap 8 to 11), and so on. The order in which the configurations occurred in each scenario are described in Figure 5.7(b). Comparing both scenarios, *Scenario B* presents small changes in the environment. Particularly, we only alter from the configuration with all doors closed (E_0) to two configurations with a single open door for each corridor (E_{B1} and E_{B2}). *Scenario C* presents bigger changes in the environment. We alter from the closed doors configuration (E_0) to other five configurations with multiple doors opening in the same corridor or opening near corners (E_{C1} to E_{C5}).

5.3.2 Analysis of Results

In a first moment, we compared the performance of single level runs against two-levels runs. In the single level runs, we just varied the minimum size of grams for accepting matches, ϵ_g, from 1 to 10 words. We only varied ϵ_g until 10, because above this value

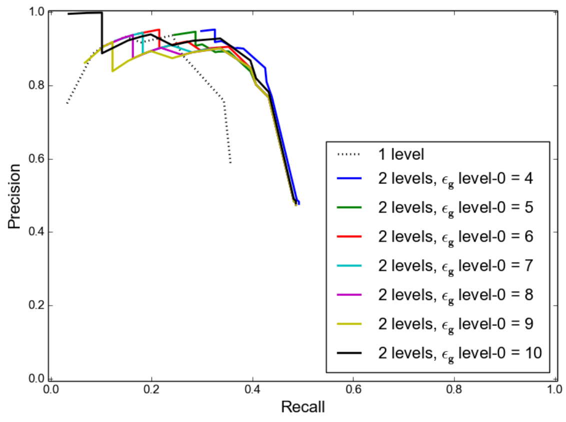
Figure 5.8: Examples of environment configurations. (a) Basic configuration used in both scenarios. (b)-(c) Configurations used in *Scenario B*. (d)-(h) Configurations used in *Scenario C*.



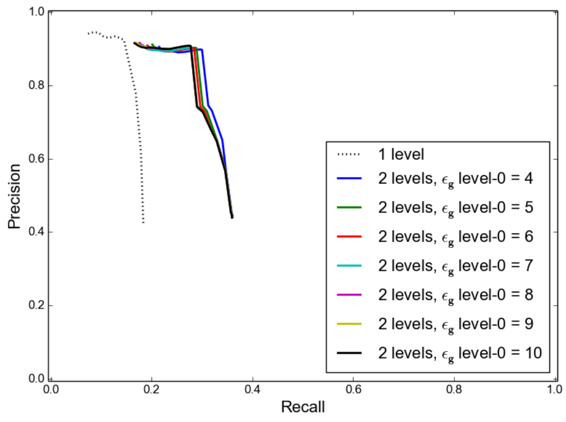
(a) E_0 (b) E_{B1} (c) E_{B2} (d) E_{C1}

(e) E_{C2} (f) E_{C3} (g) E_{C4} (h) E_{C5}

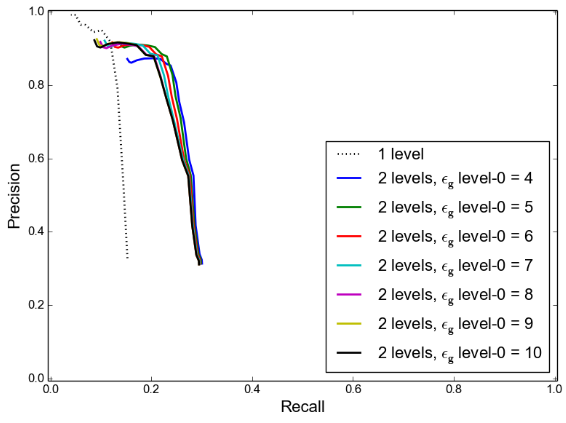
Figure 5.9: *Precision* $\times$ *Recall* plots obtained in each scenario for the single level approach and for two-levels approaches fixing the threshold of level-0 and varying the thresholds of level-1.



(a) *Scenario A*



(b) *Scenario B*



(c) *Scenario C*

the recall was near zero. In the two-levels runs, we fixed the threshold for the first level in different configurations (starting at $\epsilon_{\mathcal{g}} = 4$, because below this value the precision was very poor) and varied the threshold for the second level (from 1 to 18). Figure 5.9 shows plots of precision $\times$ recall for the three scenarios. Given that our search for matches only happens in the same direction of the robot trajectory, we decided to consider in the recall calculation only matches that could be obtained in the same direction. All results show that, with similar precision, the recall of multi-level approaches (solid lines) is higher than the recall of single level approaches (dotted lines).

Analyzing the individual results, all strategies obtained the highest recalls in *Scenario A*. This is somewhat expected because *A* is the shortest scenario, meaning that the main variations are caused by people walking, which are small in comparison to opening and closing doors. On the other hand, the lowest recalls were obtained in *Scenario C*. This was also expected because *C* is the most dynamic scenario, and it has situations that our current algorithm does not cover well, such as density variations occurring near corners. Making an analogy to Figure 5.5, a variation occurring near corners do not produce a density bump in the middle of a homogeneous region, but a density bump over another existing density bump, which is more difficult to detect.

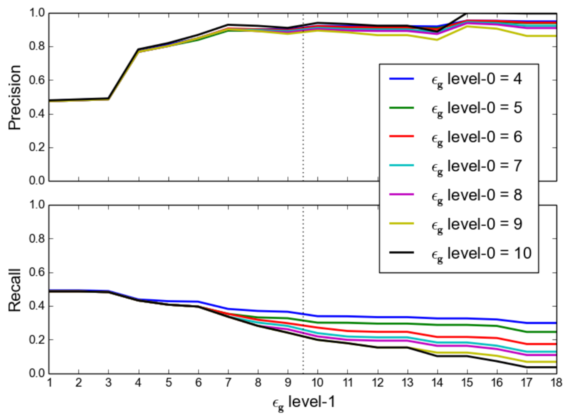
An aspect shared by all results is that the precision drops drastically when the acceptance threshold, i.e. the size of the n-grams required to accept matches, becomes too low. Figure 5.10 shows separated plots of precision and recall in relation to the the minimum size of n-grams to accept matches in *level-1* of a two-level approach (while fixing the acceptance threshold of *level-0*). We observe that increasing the size of n-grams the precision improves significantly only until some point (e.g. around $n = 9$ as shown by the vertical dotted lines). After that, the precision stays basically the same and the recall decreases. Based on these results, we can say that $n = 9$ is a good acceptance threshold for the three scenarios.

Finally, we evaluate the running time of the proposed method. One of the main advantages of the approach presented in last chapter is its high speed to compute place recognition in all trajectory. As expected, the use of multiple levels of words to improve recall makes the approach slower. Table 5.1 shows the comparison among ourthree configurations of our method[3] – a single level run, a two-level run and a three-level run – and ICP[4]. Each level that is added, reduces the speed of the method, however, even with three levels of words the method is faster than a traditional point matching technique such as ICP with few points. Comparing the results of precision and recall, each added level improves the recall with little damage to the precision. In relation to ICP, the proposed method is able to obtain much higher precision, because it can match very long segments of the trajectory in an efficient way.

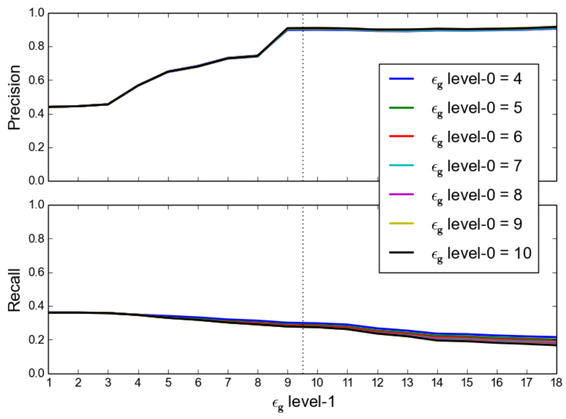
[3]The acceptance thresholds were set to 7 at *level-0*, 10 at *level-1* and 13 at *level-2*. We have chosen thresholds that increase with the level number, because we are usually able to detect larger matches when using more levels. In other words, if we use one threshold for all levels, the number of false positives will be much higher in the multi-level approaches.

[4]ICP was performed using 150 points extracted from regions created at each 20 steps

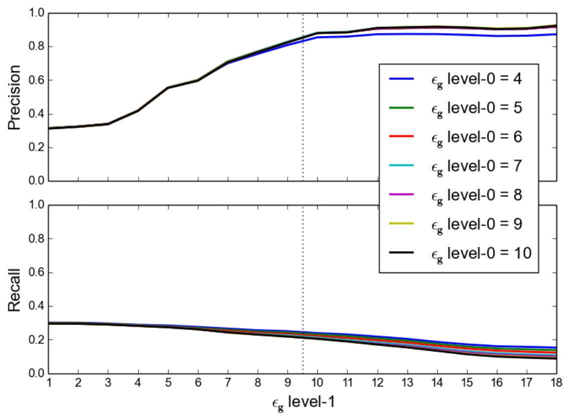
Figure 5.10: Results of *precision* (top) and *recall* (bottom) in relation to the minimum size of n-grams to accept matches in *level-1* of a two-level approach. A threshold above $n = 9$ for *level-1* (indicated by the vertical dotted line) seems good for these scenarios, given that with such thresholds the precision stays constant and the recall gradually decreases.



(a) *Scenario A*



(b) *Scenario B*



(c) *Scenario C*

Table 5.1: Comparisons of precision, recall and average time (to process one lap in the environment).

	Scenario A		**Scenario B**		**Scenario C**		
	Precision	Recall	Precision	Recall	Precision	Recall	Time(s)
1-level	0.905	0.103	0.931	0.106	0.964	0.070	2.5
2-levels	0.910	0.238	0.903	0.274	0.912	0.169	11
3-levels	0.935	0.346	0.890	0.308	0.913	0.188	38
ICP	0.048	0.040	0.047	0.011	0.219	0.040	96

5.4 Related Work

The solution of the SLAM problem requires a precise loop closing strategy (i.e. the so-called front-end of the SLAM system), which serves as basis for the map optimization (i.e the back-end of the SLAM system). Erroneous associations between places lead to drastically wrong results in the optimization process, therefore they must be avoided in all circumstances. However, the incorrect rejection of proper associations lead to less accuracy (CADENA et al., 2016). The difficulty of the problem increases when the robot is operating in long-term, due to the presence of unmodeled dynamics, such as semi-static objects (e.g. furniture) in indoor environments or seasonal changes in outdoor environments.

During the last years, various approaches for place recognition in dynamic scenarios have been presented in the literature, mostly based on computer vision and using a wide range of different strategies to find matches, such as template matching, feature extraction, Bag of Words, etc (CUMMINS; NEWMAN, 2008; GALVEZ-LÓPEZ; TARDOS, 2012; MILFORD; WYETH, 2012). On the other hand, place recognition based only on laser measurements suffers from the high level of ambiguity associated with the sensors information. For instance, a robot that is moving inside a structured environment can obtain the same information in multiple occasions at different places, and thus the chance of producing false matches is large. In cases like this, the only way to disambiguate similar regions is by considering sequential information. Many works show that exploring sequences of observations during the place recognition process leads to better results than matching single observations (BOSSE; ZLOT, 2008; MILFORD; WYETH, 2012).

Some of the earliest long-term SLAM strategies, proposed by Biber and Duckett (BIBER; DUCKETT, 2005; BIBER; DUCKETT, 2009), use local maps that are sampled at different time scales, in order to create maps with short and long-term objects. Such type of approach is able to cover variations from moving objects to structural changes in the environment in time scales of weeks. However, their solution requires large amounts of memory and only operates over an initial static map, which means that cannot incorporate new map sections. Konolige and Bowman (KONOLIGE; BOWMAN, 2009) also present a strategy with multiple representations of dynamic environments, but are able to deal with incremental map additions and large changes in the environment. Tipaldi et al. (TIPALDI et al., 2011; TIPALDI et al., 2012) propose using a single dynamic occupancy grid, instead of multiple maps, to deal with changes in the environment. Their method, which represents the grid cells (states of the environment) with a hidden Markov model, is able to considerably minimize the memory requirements, in comparison to the previous methods. A different way of facing the long-term SLAM problem, as proposed by Krajník et al. (KRAJNIK et al., 2014), is modeling the spatio-temporal dynamics of an environment by its frequency spectrum, which allows the efficient detection of regularly occurring process (e.g. opening/closing doors). Their approach is specially good for predicting future states of a well-behaved environment with high accuracy.

A major issue of long-term SLAM is that the size of the state to be estimated (map and robot trajectory) can grow unbounded, given that the robot operates for extended periods of time in a continuous exploration process. One way to reduce the size growth of the long-term SLAM problem is through sparsification methods, which improve computational efficiency and reduce memory requirements, at the cost of information loss (CADENA et al., 2016). Johannson et al. (JOHANNSSON et al., 2013) is a graph-based method that avoids indefinite growth by adding extra constraints to the existing nodes, instead of adding new nodes with redundant poses to the graph. Walcott-Bryant et al. (WALCOTT et al., 2012) propose the dynamic pose graph, which keeps nodes associated to dynamic local grid maps. Regularly, their method removes "inactive" nodes, i.e. associated to regions whose occupancy information is outdated. An assumption of their method is that the robot starts and ends in known points of the environment to avoid alignment errors.

Our approach deals with the size problem of long-term SLAM by compressing all information obtained by the robot sensors into simple words. Like some other works (BIBER; DUCKETT, 2005; KONOLIGE; BOWMAN, 2009; WALCOTT et al., 2012), we handle the dynamics of the environment by generating multiple representations of the same place, which in our case are described by words. Lastly, we do not make assumptions about initial robot poses of each test section.

5.5 Summary

In this chapter we propose a strategy for long-term place recognition by matching sequences of words built from spatial density information. We can detect changes in the environment by creating multiple words associated to the same region. That means, given a density variation observed in the robot trajectory, our method builds a word to represent such variation and build other words to represent what the robot should be seeing if such variation did not exist. Experiments were made in real and simulated scenarios of an indoor environment, and have demonstrated that the multi-level approach obtained improvements in the matching results when compared to the single-level approach.

The main contributions of the approach are:

(i) A novel representation of dynamic regions associating words of spatial density in multiple levels;

(ii) A fast strategy for finding matches in sequences of multi-level words;

(iii) A fine-tuning adjustment of matches using raw kernel density estimates of the free space.

We have observed that detecting changes in the environment near places with high variations of densities is still a problem for the method. At this moment, the method only deals with variations that are preceded and succeeded by densities of the same class. Thus, we are studying ways of generating alternative words for dynamic objects in situations like that.

Finally, our current method only searches for matches in the same direction that was visited by the robot, since it is based on n-grams, i.e. ordered sequences of words. As future work, we also want to generate reversed words, and find matches using such information.

6 ANALYSIS OF FSD-BASED WORDS AND THEIR USE FOR SEMANTIC MAPPING

6.1 Introduction

Robots must have good localization and mapping capabilities in order to operate alongside people in real-world environments. Usually, most of the mapping approaches in the literature focus on the construction of consistent metric maps. In other words, they represent the environment using specific metric representations, such as occupancy grids and sets of landmarks. While such type of information can be easily handled by robots, the same cannot be said in relation to humans. In fact, the interaction between humans and robots is much improved if the robots are able to deal with semantic information associated to the environment (KOSTAVELIS; GASTERATOS, 2015).

A person analyzing indoor environments, such as the ones investigated in this thesis, would typically divide them into regions like rooms, corridors and corners. This type of qualitative description of the environment can be used in human-robot interaction, for instance, to facilitate the assignment of tasks. In general, it is much more intuitive for a human to tell a robot to "*go forward in the corridor and turn right at the corner*" than saying "*go to pose* $x = 20m$, $y = 5m$, $\theta = 90°$". Notwithstanding, for this truly works, the robot must be able to understand this type of information which is done by the process of semantic mapping.

Our main proposal in this thesis is a technique that divides the environment into regions and describe them with words. Then, we search for matches of sequences of words to solve the place recognition problem. Methods of place recognition are similar to methods of semantic mapping, still there is an important difference between them. In place recognition, the goal is to find matches between regions, which can be done with the aid of a wide range of techniques, such as scan-matching (CHEN; WANG, 2006), image alignment (MILFORD; WYETH, 2012), feature-matching (CUMMINS; NEWMAN, 2009; MUR-ARTAL; MONTIEL; TARDÓS, 2015), among others. It does not matter neither how the information is represented, nor whether there is any intuitive meaning behind, it only matters that the correspondences are correct. Semantic mapping approaches usually include some form of place recognition, nevertheless, they must also include place categorization, which means the capability of classifying places using intuitive labels (MOZOS, 2010). Among the approaches for semantic mapping, many perform classification of places by manually defining models to deal with sensors information, for example by defining that corridors are equivalent to two parallel lines of obstacles, (NÜCHTER et al., 2003; GALINDO et al., 2005); while many perform classification using machine learning techniques such as hidden markov models (HMM) (STACHNISS; MOZOS; BURGARD, 2006), support vector machines (SVM) (WOLF; SUKHATME, 2008), Adaboost (ROTTMANN et al., 2005; MOZOS; BURGARD, 2006), etc.

This chapter presents a brief study of our method in the light of semantic mapping. We analyze how is the behavior of the FSD-based words associated to different types of regions, such as corridors, corners and bifurcations. We check if such behavior is consistent not only in simulated scenarios, but also in real scenarios filled with dynamic obstacles, apertures (e.g. open doors) and other unconventional features (e.g curved corridors). Later, based on this study, we propose a simple strategy to attribute semantic meanings to the words created by our method.

6.2 An in-depth study of FSD-based words generated in different environments

6.2.1 Ground-truth classification of several scenarios

Before starting our analysis, we present the environments to be studied in Figures 6.1, 6.2 and 6.3. For each scenario we present two types of maps: one showing the robot path and one showing the ground-truth of semantic division. We made the semantic division of each environment by manually coloring regions of the same type. In our study, we are analyzing five different types of regions that can be found in indoor environments:

- **corridors**: narrow passageways, delimited by two parallel walls, with apertures/doors for adjacent rooms;
- **corners**: segments of corridors with abrupt change of direction (usually, around 90 degrees);
- **bifurcations**: regions connecting three or more passages, such as T-junctions (when a corridor ends in the middle of another corridor) or crossroads (when there is a crossing of two orthogonal corridors);
- **dead-ends or rooms**: small regions surrounded by walls with, usually, only one access area;

- **halls**: large entrances or rooms.

Six indoor scenarios were selected for experiments: three simulated and three real. Figure 6.1 shows the three simulated scenarios – S1, S2, and S3. We created those scenarios using the MobileSim Simulator from Adept MobileRobots, and actually, they are some of the scenarios used throughout this thesis: S1 and S2 are static environments analyzed in Chapters 3 and 4, and S3 is a semi-static environment analyzed in Chapter 5. Figures 6.2 and 6.3 show three real scenarios – R1, R2, and R3. Those are public datasets[1] widely used by the SLAM community. Table 6.1 presents information about these datasets.

Table 6.1: Information about the datasets of real scenarios.

Scenario	R1	**Original Name**	ACES3, Austin
Provider	Patrick Beeson	**Date**	07 May 2004
Description	ACES building on Univ. Texas Austin campus		
Scenario	R2	**Original Name**	Intel
Provider	Dirk Hähnel and Dieter Fox	**Date**	13 May 2003
Description	Interior of the Intel Research Lab in Seattle		
Scenario	R3	**Original Name**	MIT Infinite Corridor
Provider	Mike Bosse and John Leonard	**Date**	11 Sep 2002
Description	Interior of MIT Killian Court		

All observations made by the robot in each scenario were classified according to the ground-truth. This classification was made by checking the robot positions, from where the observations were taken, in the maps of semantic divisions that are presented in Figures 6.1, 6.2 and 6.3. The distribution of all observations[2] in the five semantic classes – in absolute number of observations (shown in black) and percentage of total observations in each scenario (shown in gray) – is presented in Table 6.2, . In terms of number of observations made by the robot, the simulated scenarios – particularly S1 and S3 – are the longest ones. This happens because, differently from the real scenarios, the simulated scenarios were aimed for long-term tests, which implies in extensive revisiting of known areas. Nonetheless, the real scenario R3 is also very long because it is associated to an environment much larger than all the other environments (note the different scales of Figure 6.3 and Figures 6.1 and 6.2).

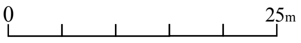
[1]The three datasets of real environments were obtained from the Photogrammetry Lab website from University of Bonn, headed by prof. Cyrill Stachniss, <http://www.ipb.uni-bonn.de/datasets/>, and were originally available in the Robotics Data Set Repository (Radish) (HOWARD; ROY, 2003). Thanks go to Beeson, Hahnel, Fox, Bosse, Leonard and Stachniss for providing this data.

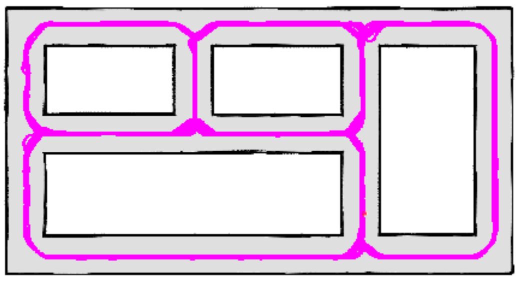
[2]In all scenarios, the sampling rate is 10 observations by $1m$ traveled by the robot.

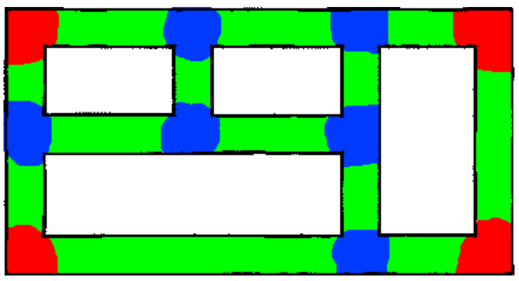
Figure 6.1: Simulated scenarios S1, S2 and S3.

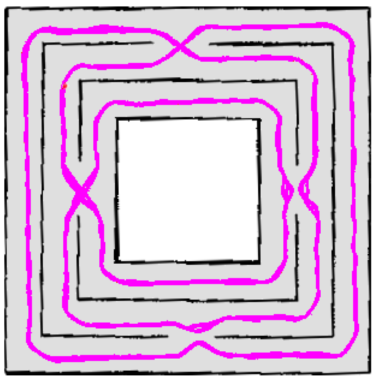
CAPTION

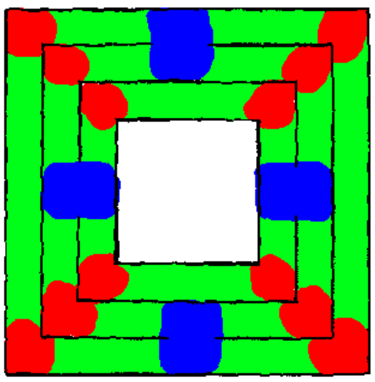
— robot path	▨ free-space	🟦 bifurcation	🟥 corner	🟩 corridor	🟨 dead-end *or* room

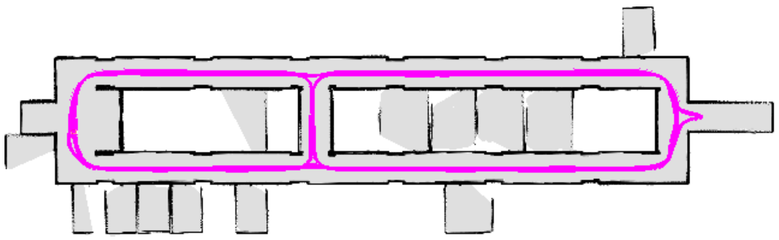



(a) Scenario S1: map and robot path


(b) Scenario S1: semantic division


(c) Scenario S2: map and robot path


(d) Scenario S2: semantic division


(e) Scenario S3: map and robot path

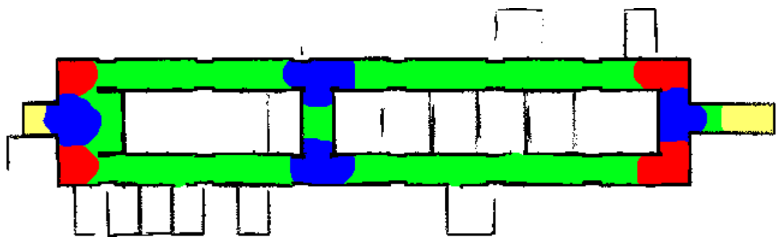
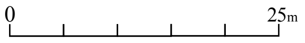
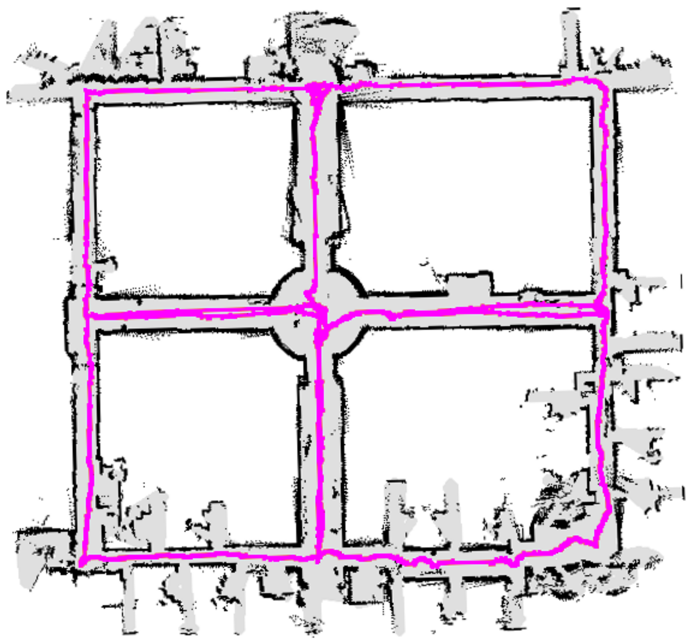

(f) Scenario S3: semantic division

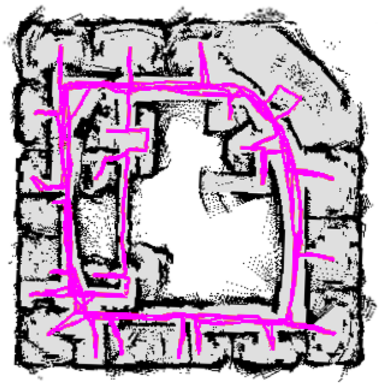
Figure 6.2: Real scenarios R1 and R2.

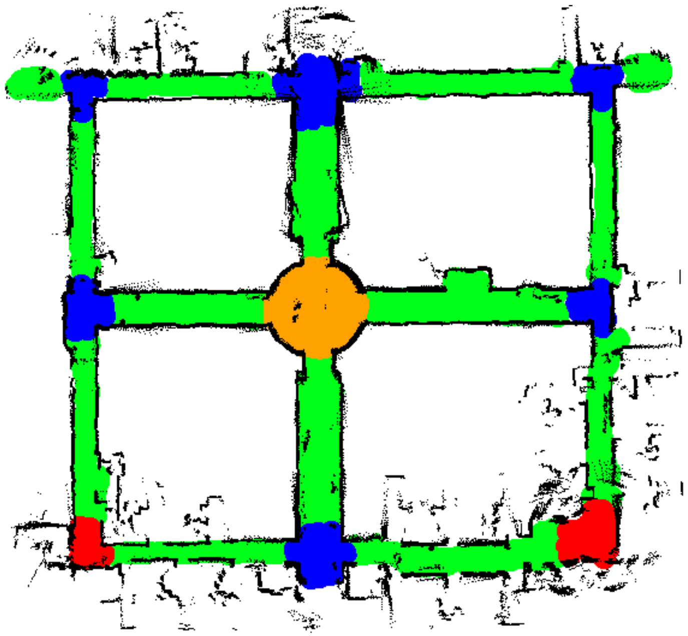
CAPTION

— robot path ▨ free-space 🟦 bifurcation 🟥 corner 🟩 corridor 🟨 dead-end *or* room 🟧 hall




(a) Scenario R1: map and robot path


(b) Scenario R2: map and robot path


(c) Scenario R1: semantic division

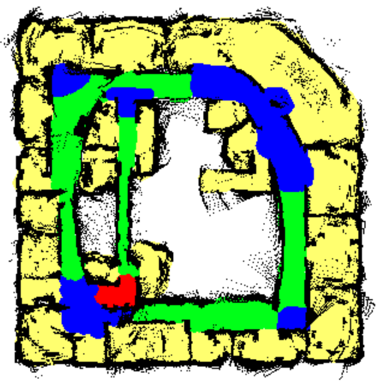
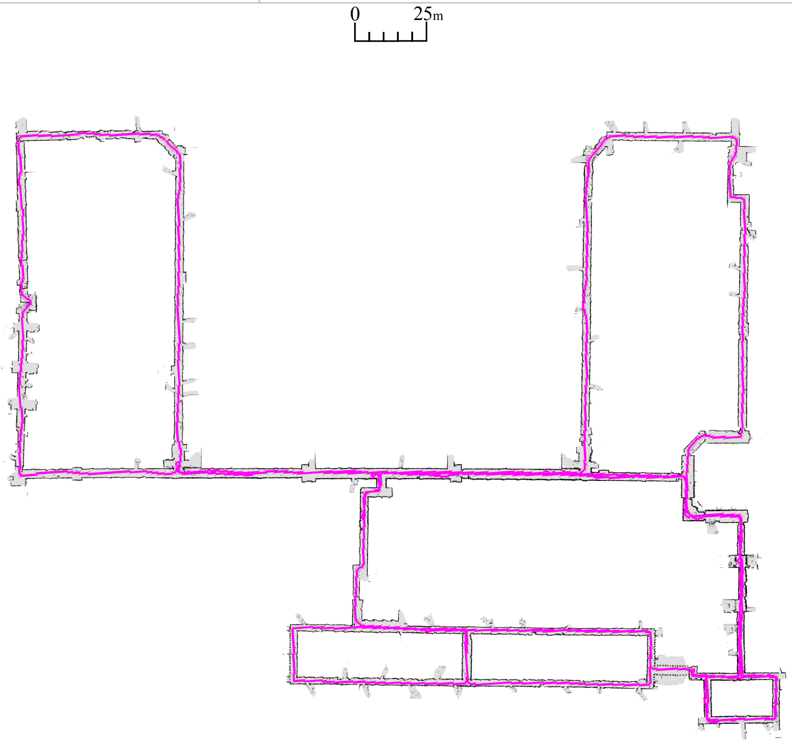

(d) Scenario R2: semantic division

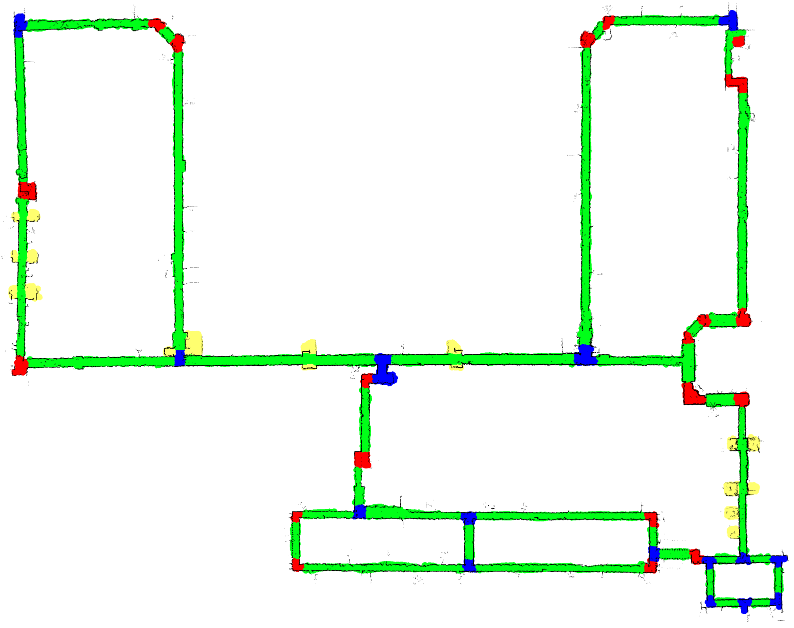
Figure 6.3: Real scenario R3.

CAPTION

— robot path ▨ free-space 🟦 bifurcation 🟥 corner 🟩 corridor 🟨 dead-end *or* room



(a) Scenario R3: map and robot path



(b) Scenario R3: semantic division

Table 6.2: Semantic division of all observations according to the ground-truth

Scenarios	bifurcation		corner		corridor		dead-end		hall		TOTAL
	#	%	#	%	#	%	#	%	#	%	#
S1	7008	*25.3*	3794	*13.7*	16888	*61.0*	0	*0.0*	0	*0.0*	27690
S2	1202	*16.4*	1346	*18.4*	4773	*65.2*	0	*0.0*	0	*0.0*	7321
S3	7537	*19.3*	3825	*9.8*	27781	*71.0*	0	*0.0*	0	*0.0*	39143
R1	543	*26.9*	81	*4.0*	1282	*63.6*	0	*0.0*	111	*5.5*	2017
R2	1150	*30.9*	59	*1.6*	1851	*49.8*	658	*17.7*	0	*0.0*	3718
R3	1968	*12.7*	1577	*10.2*	11896	*77.0*	0	*0.0*	0	*0.0*	15441

In all scenarios, the robot passes over three types of regions: corridors, corners and bifurcations. Corridors are the most common part of all environments, reaching 77.0% of the visited space in scenario R3. Dead-ends/rooms are only visited by the robot in scenario R2, corresponding to 17.7% of the visited space. Finally, there is only one visited hall in all environments, which is located in scenario R1, corresponding to 5.5% of the visited space.

Table 6.3 shows the ground-truth of expected words obtained from the robot path in each one of the six scenarios. This distribution of words is made considering repetitions, i.e., we are not excluding possible multiple instances of the same word generated by revisits of the same place.

Table 6.3: Expected words according to the ground-truth

Scenarios	bifurcation		corner		corridor		dead-end		hall		TOTAL
	#	%	#	%	#	%	#	%	#	%	#
S1	231	*33.2*	116	*16.7*	348	*50.1*	0	*0.0*	0	*0.0*	695
S2	38	*19.9*	57	*29.8*	96	*50.3*	0	*0.0*	0	*0.0*	191
S3	185	*32.5*	159	*27.9*	225	*39.5*	0	*0.0*	0	*0.0*	569
R1	17	*44.7*	2	*5.3*	15	*39.5*	0	*0.0*	4	*10.5*	38
R2	34	*35.1*	3	*3.1*	34	*35.1*	26	*26.8*	0	*0.0*	97
R3	39	*30.5*	29	*22.7*	60	*46.9*	0	*0.0*	0	*0.0*	128

The goal of our method for building words is to create one word for each path segment that passes over a contiguous region of the environment. For example, if the robot starts in a bifurcation, traverses a corridor, and turns in a corner, the best outcome should be a sequence of three words associated to bifurcation, corridor and corner. In practice, it is not easy to exactly predict the expected sequence of words due to variations associated to dynamic objects, or in the robot path, etc. Still, there are some distinguishable characteristics that arise in specific types of regions, which can be used to better build words according to semantic concepts. Next, we investigate each type of region in terms of the three syllables that compose our proposed words: free-space density, length of the region, variation in the robot orientation traversing that region.

6.2.2 Syllable `d`: the characteristics of free-space density in different places

As shown in Chapter 4, we perceived that in well-behaved scenarios, such as S1, the densities of corridors, corners and bifurcations are different. Corners tend to have smaller density values than bifurcations, and larger density values than corridors. However, this is only valid considering constant width for all corridors, corners and bifurcations, which many times is not the case.

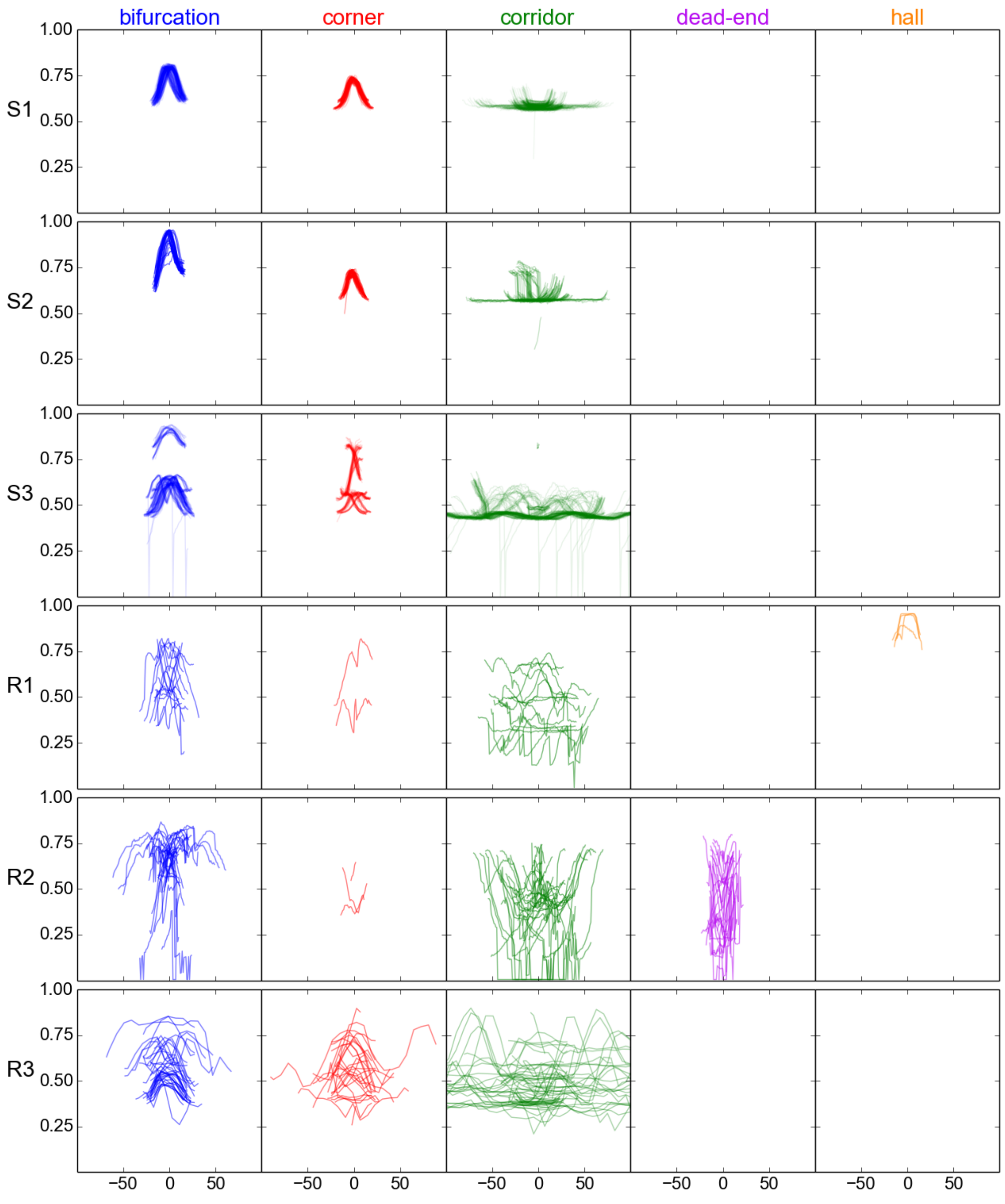
Figure 6.4 shows the variation of free-space density in each type of region (vertical division) for all six scenarios (horizontal division). Each colored line represents an instance of a word, where the x-axis indicates the sequence of observations composing the word and the y-axis indicates the density associated to each observation. All words are centered in 0 regarding the x-axis, which means that a line that goes from -50 to 50 corresponds to a sequence of 101 observations. This center alignment of words is made in the plot because the information of what happens in the beginning, the middle and the end of the regions may help us to classify them.

In terms of density variation, one of the most important information that we can extract from Figure 6.4 is that usually the free-space density in bifurcations and corners varies in a predictable way: it starts with a given value, than it climbs to some maximum value (which tends to occur in the middle of the region), and finishes falling back to the original value. In contrast, the free-space density in corridors tends to stay flat in the majority of the region, only varying a little in both ends (i.e., in the transition to other regions).

In terms of absolute density values, we are not able to say much about bifurcation, corners, corridors and dead-ends. However, we can extract an important information about halls in scenario R1. Given that halls are large empty spaces, they tend to be quite predictable because the free-space density stays near 1.0 (i.e. the maximum possible value) in this type of environment.

Summing up, based only on the density information, we can try to differentiate corridors from bifurcations and corners, and also to detect halls. Still, in some scenarios the density variation may be too noisy to infer something with good accuracy. For example, looking back at scenarios R1 and R2 in Figure 6.2, we can see that there are numerous open doors in the corridors visited by the robot, and all these doors distort the free-space density. Thus, more information sources are needed to distinguish all regions.

Figure 6.4: Variation of the free-space density in the robot position (y-axis) considering the sequence of observations (x-axis) that compose each word.



6.2.3 Syllable s: the characteristics of region length in different places

We can also verify in Figure 6.4, by looking at the x-axis, if some reliable information can be obtained from the length of different types of regions, i.e. from the size of the sequences of observations obtained in each region. Usually, the longest regions are always associated to corridors. In fact, with the exception of a few outliers in scenarios R2 and R3, all words with region length larger than 100 observations ($\sim 10m$) correspond to corridors. However, short words cannot be classified based only on their lengths, given that they can be associated to practically all types of regions, such as bifurcations, corners, dead-ends, or even small corridors.

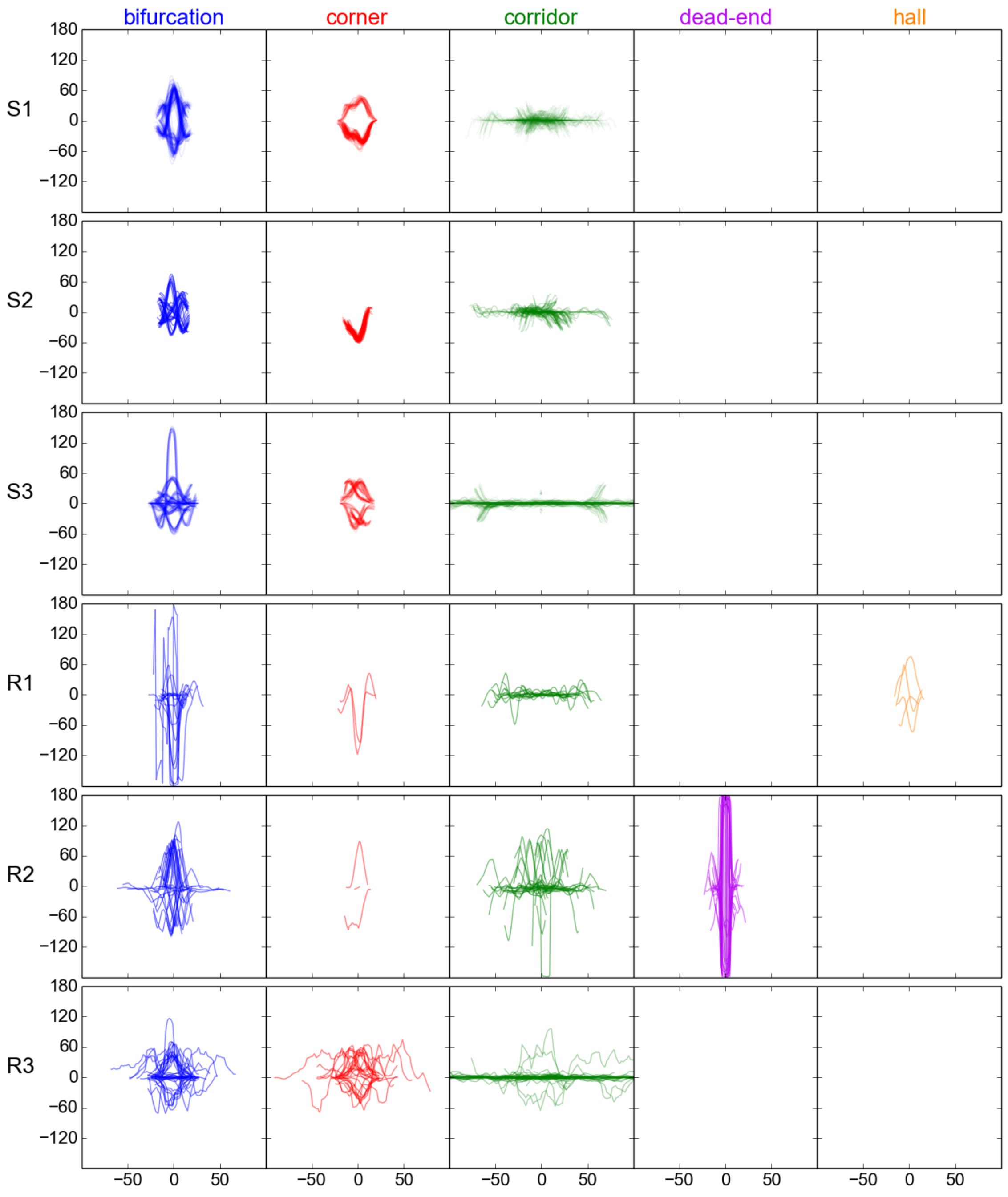
6.2.4 Syllable a: the characteristics of angle variation in different places

The variation of the robot orientation in each region (vertical division) for the six scenarios (horizontal division) can be analyzed in Figure 6.5. Analogous to Figure 6.4, each instance of a word is represented by a colored line, where the x-axis indicates the sequence of observations composing the word and the y-axis indicates the variation, in degrees, of the robot orientation. This variation of orientation is computed following the definition of the syllable a presented in Chapter 4, but now we are computing it at each robot pose (considering a range of 10 poses), instead of only computing it in the middle pose of each region[3].

The behavior of the robot orientation clearly changes from corridors to corners. While in corridors the robot orientation stays fixed most of the time, in corners the orientation always varies. One important point is that the orientation information in corridors is much more stable than the information of free-space density, since apertures in the environment does not affect this type of information. Bifurcations are a problem to classify because the robot orientation changes in different ways depending on the trajectory made by the robot. Many times they will look the same as corners, and other times they will look similar to corridors (although, in this case, they are easy to differentiate based on density information). Regarding other types of regions, the information about robot orientation is very good for the identification of dead-ends (and rooms). As shown in the plot associated to scenario R2, the orientation drastically varies when the robot visits such places, which can be explained by a simple fact: the robot enters the room in a given direction and, usually, turns 180 degrees to leave the room.

[3]As a recap, the variation is given by the difference of the angle between median and final poses of the sequence of observations, and the angle between initial and median poses. For more details, see the explanation of the Algorithm 4.3 in Section 4.2.2.

Figure 6.5: Variation, in degrees, of the robot orientation (y-axis) considering the sequence of observations (x-axis) that compose each word.



6.2.5 Summary of the analysis

After the analysis of the three types of information associated to the words built by our method, we are able to extract some general aspects about each type of region:

- Corridors have low variation of free-space density and low variation of robot orientation. They can also be very long, differently from all the other regions;

- Corners have high variation of free-space density (but, usually, the density returns to the original value) and high variation of robot orientation;
- Bifurcations have high variation of free-space density (just like corners). Unfortunately, not much can be stated in terms of robot orientation or region length;
- Dead-ends (or rooms) have very high variation of robot orientation;
- Halls have very high peaks of free-space density.

Corridors, dead-ends and halls have some unique characteristics (e.g. low variation of free-space density, or very high peaks of free-space density), and thus, they are the most distinguishable regions in the tested scenarios. On the other hand, corners and bifurcations share many characteristics. For instance, if the robot is turning $90°$ and we are detecting a high variation of free-space density, there is a high chance that the robot is in a corner or bifurcation, but we will not be able to easily decide between the two.

It was shown in previous chapters, that the information of the three syllables `d`, `s`, and `a` are suitable for the problem of place recognition. However, it is important to note that in such case we are only interested in finding matches of existing words, without mattering what they mean. In this section, we can see that the same information of the three syllables is not sufficient for a proper semantic classification of all regions. Still, as presented in some of the most prominent works on semantic mapping (MOZOS, 2010; ROTTMANN et al., 2005; STACHNISS; MOZOS; BURGARD, 2006), there are plenty of other types of information that can be used to improve disambiguation, such as the shape or perimeter of the region measured by the robot, the number of gaps in the sensors measurements, etc. Since we are already using a circular kernel to compute the free-space density, we decided to briefly investigate another kernel-based feature: the number of frontiers of free-space in the borders of the kernel.

6.3 Investigating the number of frontiers of free-space

As described in Chapter 3, the free-space density is computed with a flood-fill strategy applied inside a circular kernel. With some minor modifications to the original algorithm, we can extract the cells at the border of the kernel. Those cells can be used to compute some metrics such as the perimeter of free-space, or the numbers of frontiers of free-space. Given that we want to find some environment feature that helps us to differentiate corners from bifurcations, we have preferred to analyze the number of frontiers of free-space.

Algorithm 6.1 describes the strategy to compute the number, $\mathtt{f}$, of frontiers of free-space in the boundaries of a kernel K_h centered at cell $\boldsymbol{m}_0$. The first part of the algorithm, which is quite similar to Algorithm 3.1 for computing the free-space density, consists in finding the cells at the border of the kernel, and adding them to a list B (lines 1-11). A flood-fill technique is used to expand all free-space cells whose distance to the center cell, d_i, is smaller than the kernel bandwidth h (lines 5-7). The border cells are the reachable free-space cells slightly outside the kernel boundaries (lines 10-11).

The second part of the algorithm is the segmentation of the border cells in contiguous frontiers (lines 12-23). Each frontier F is built by initially removing from B one cell, $\boldsymbol{m}_i$ (line 14), and then recursively removing from B all neighbors that are connected to $\boldsymbol{m}_i$ (lines 17-21). The frontier is complete when no more cells can be added to F. When this happens, a new frontier starts to be generated from another random cell in B. We only keep frontiers with size larger than a given threshold[4] $\epsilon_{\mathtt{f}}$ (lines 22-23), because tiny

[4]We empirically defined $\epsilon_{\mathtt{f}}$ as 5% of the kernel perimeter.

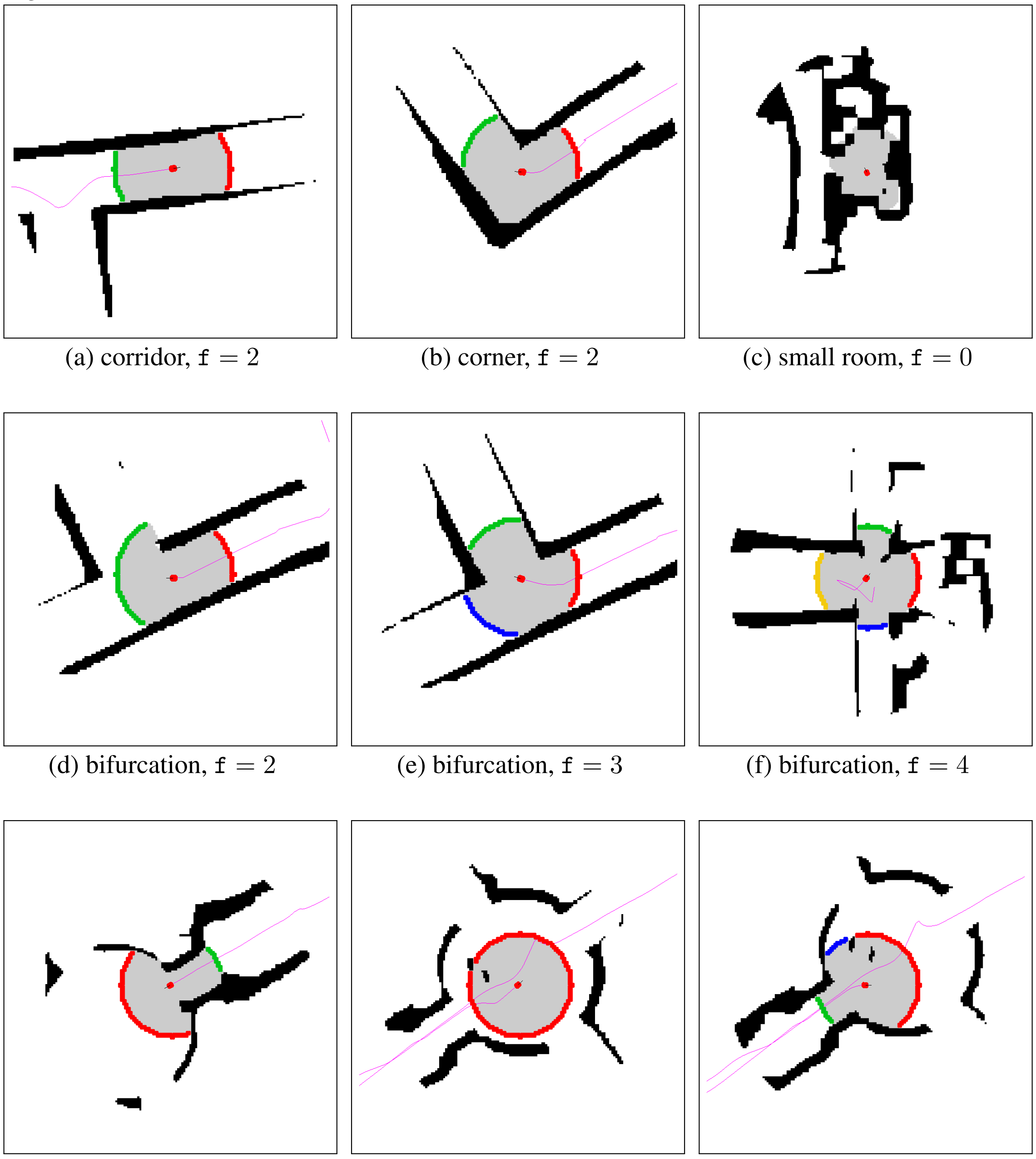
Algorithm 6.1: Compute number of free-space frontiers

```
Input: $\boldsymbol{m}_0, \boldsymbol{m}, K_h, \epsilon_{\mathtt{f}}$
Output: $\mathtt{f}$

1  $B = [\ ]$
2  $Q = [\boldsymbol{m}_0]$
3  while $Q \neq [\ ]$ do
4      $\boldsymbol{m}_i = pop\_front(Q)$
5      if $\boldsymbol{m}_i$ is free-space then
6          $d_i = \sqrt{(x(\boldsymbol{m}_i) - x(\boldsymbol{m}_0))^2 + (y(\boldsymbol{m}_i) - y(\boldsymbol{m}_0))^2}$
7          if $d_i < h$ then
8              foreach (non processed) neighbor $\boldsymbol{m}_k$ of $\boldsymbol{m}_i$ do
9                  $push\_back(\boldsymbol{m}_k, Q)$
10         else
11             $push\_back(\boldsymbol{m}_i, B)$

12 $\mathtt{f} = 0$
13 while $B \neq [\ ]$ do
14     $\boldsymbol{m}_i = pop\_front(B)$
15     $F = [\ ]$
16     $Q = [\boldsymbol{m}_i]$
17     while $Q \neq [\ ]$ do
18         $\boldsymbol{m}_j = pop\_front(Q)$
19         $push\_back(\boldsymbol{m}_j, F)$
20         foreach existing neighbor $\boldsymbol{m}_k$ of $\boldsymbol{m}_j$ in $B$ do
21             $push\_back(\boldsymbol{m}_k, Q)$
22     if $size(F) > \epsilon_{\mathtt{f}}$ then
23         $\mathtt{f} \mathrel{+}= 1$

24 return $\mathtt{f}$
```

frontiers are generally the result of holes in the grid map.

Figure 6.6 presents some examples of how the number of frontiers of free-space varies in different types of regions. In corridors and corners, as shown in (a) and (b), there are usually two frontiers (one in the front, and one behind the robot). However, if the region has moderate/large apertures the number of frontiers increase (in practice, such situations are viewed as bifurcations). In small rooms the number of frontiers tends to drop because, in general, there are only small apertures of free-space, as shown in (c), and our algorithm

Figure 6.6: Example of the number of frontiers of free-space, `f`, in different types of regions.



(a) corridor, `f` $= 2$ (b) corner, `f` $= 2$ (c) small room, `f` $= 0$

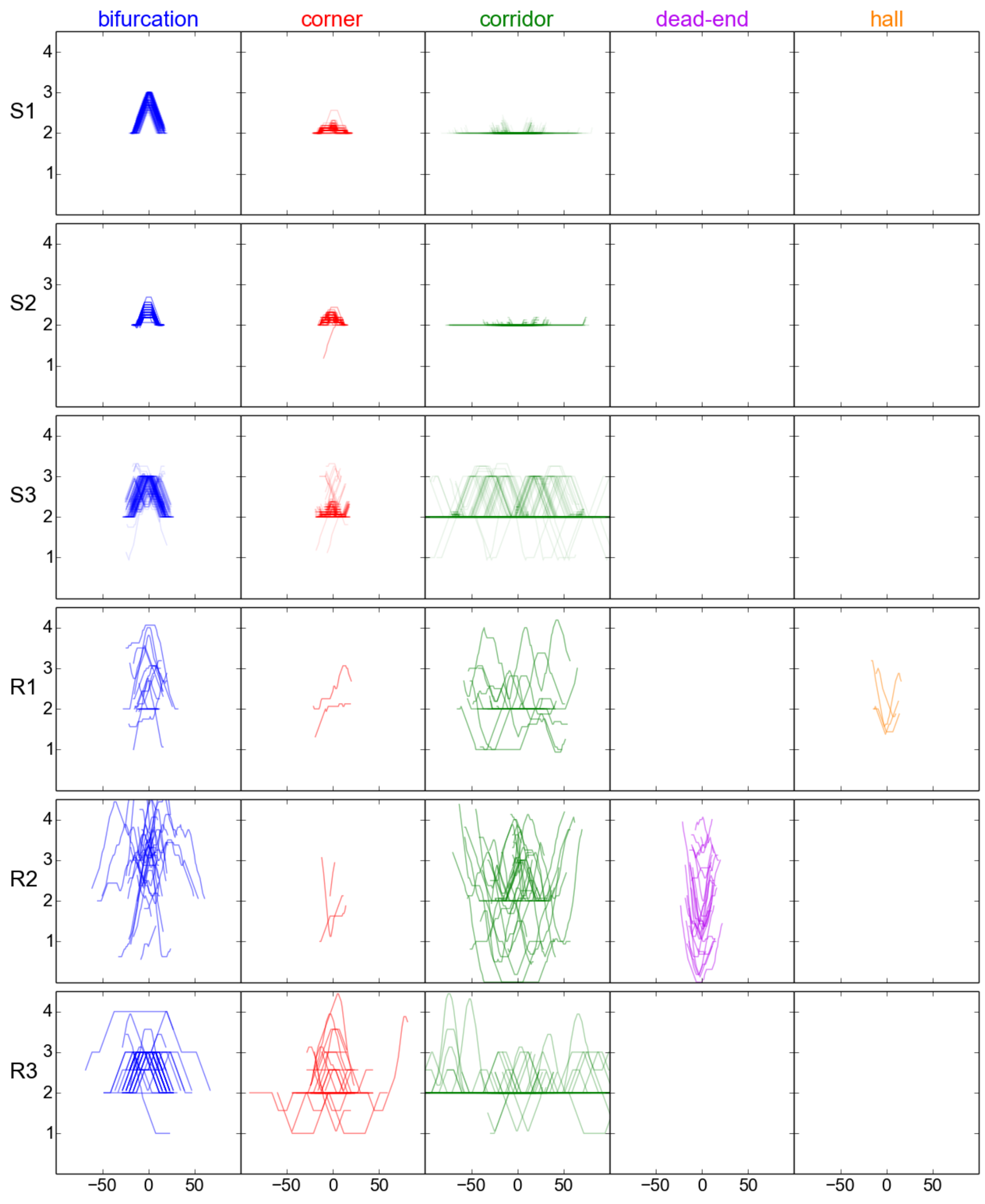
(d) bifurcation, `f` $= 2$ (e) bifurcation, `f` $= 3$ (f) bifurcation, `f` $= 4$

(g) hall, `f` $= 2$ (h) hall, `f` $= 1$ (i) hall, `f` $= 3$

discards small frontiers of free-space. Around bifurcations, the number of frontiers grows from two to three, four, or even higher, depending on the type of the region, as shown in (d)-(f). In halls, the number of frontiers tends to decrease to one, because the whole kernel around the robot enters the free-space, as shown in (g)-(h). Still, it is important to keep in mind that the presence of dynamic obstacles in all types of regions may affect this metric, as shown in (i).

We evaluated this metric in all scenarios – S1, S2, S3, R1, R2, R3 – and the results are shown in Figure 6.7. The colored lines represent instances of words, the x-axis indicates the sequence of observations composing the words and the y-axis indicates the number of free-space frontiers associated to each observation. Since this number is discrete, we apply a smoothing filter to reduce the influence of noise. Analyzing the simulated scenarios, we observe that there is always a high positive variation in the number of frontiers in bifurcations. Basically, the sequence of observations usually starts with two frontiers, climbs to three or more frontiers, and returns to two. In corners and corridors, the variation is generally much smaller. However, analyzing the real scenarios, we see that high variations may also occur in corners and, specially, in corridors, due to open doors and dynamic obstacles. This is the main reason that makes the smoothing step necessary, as well as the discard of tiny frontiers. To conclude, dead-ends and halls are usually associated to negative variations in the number of frontiers, either because the robot enters large free-space areas or because the robot is almost completely surrounded by obstacles.

Figure 6.7: Variation of the number of free-space frontiers surrounding the robot (y-axis) considering the sequence of observations (x-axis) that compose each word.



6.4 A preliminary approach based on free-space density for environments classification

Based on the analysis results presented in this chapter, we propose and evaluate a simple technique of word classification. Our strategy classifies words according to the variation of the free-space density, the maximum value of density values, the length of the region, the variation in the robot orientation and the number of free-space frontiers.

As done by many authors in the literature (MASON; MARTHI, 2012; RUSU et al., 2008; NÜCHTER et al., 2006; ALTHAUS; CHRISTENSEN, 2003), we manually define the parameters of our model. Surely, in the future, the investigation of machine learning techniques is recommended to automatically determine the best parameters and features to improve the semantic mapping (KOSTAVELIS; GASTERATOS, 2015; PRONOBIS; JENSFELT, 2012; MOZOS, 2010). Nevertheless, the main idea here is to show that we can easily add semantic information to our textual representation of observations.

The first step before classifying the places visited by the robot is the segmentation of the sequence of observations. In Chapters 4 and 5, we proposed the quantization of free-space density into classes, and the translation of sequences of densities from same class into words. This approach works well for place recognition because in such problem we are matching observations obtained inside the same environment. For example, there is no problem for place recognition if the words associated to corridors in an environment are completely different from the words associated to corridors in other environment. For semantic mapping, however, the segmentation must be made in a more clever way, i.e., we need to use some information independent from the environment. As we saw in the current chapter, the free-space density in corridors is stable, while in all the other places the density fundamentally varies. Therefore, our proposal for segmentation is based on the variation of free-space density: a segmentation happens whenever the FSD starts or stops to vary[5]. Given that in structured indoor environments, the robot generally moves between corridors and some other region, this strategy works quite well.

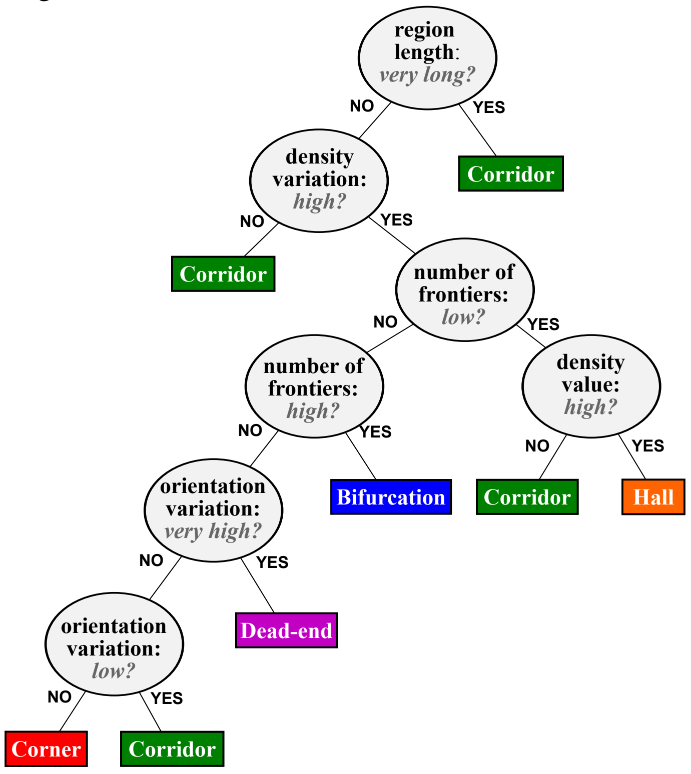
After the generation of words associated to all segments of the robot path, we perform their classification into five possible classes: bifurcations, corners, corridors, dead-ends and halls. Figure 6.8 presents the proposed strategy for semantic classification in the form of a decision tree. The (empirically defined) parameters associated to each one of the tested conditions are described in Table 6.4. First, words associated to very long regions

[5]The density values are smoothed with a mean filter to reduce the influence of noise.

Table 6.4: Parameters used in semantic classification.

Metric	Condition	Threshold
density variation	high	> 0.1
density value	high	> 0.8
number of frontiers	low	< 1.5 frontiers
number of frontiers	high	> 2.3 frontiers
orientation variation	very high	$> 120°$
orientation variation	low	$< 20°$
region length	very long	> 100 obs. $\approx 10m$

Figure 6.8: Decision tree for semantic classification.

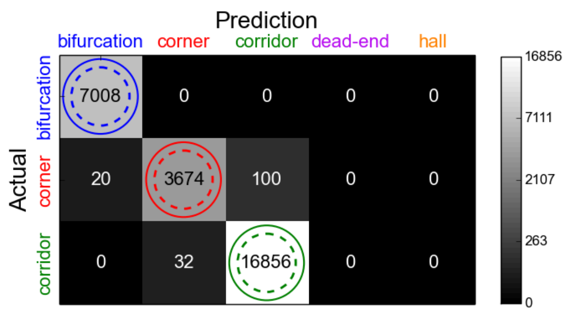


or with low density variation are classified as corridors. Next, we classify places with low number of free-space frontiers in two ways: as halls, if they reach high density values; or as corridors, otherwise. Places with high number of free-space frontiers are classified as bifurcations. Finally, places with huge variation on the robot orientation are classified as dead-ends; places with low variation are classified as corridors; and the remaining are classified as corners.

Our strategy was evaluated in three simulated scenarios (S1, S2, S3) and three real scenarios (R1, R2, R3), and the results are presented in Figures 6.9 and 6.10, respectively. For each scenario, we have computed a confusion matrix[6] that indicates the predictions made by our method in relation to the actual occurrences of each type of place. We also highlight the most predicted value among the actual occurrences of each type of place (solid circles), and the actual value that most occurred among the predicted occurrences of each type of place (dashed circle). Additionally, for each scenario we present a table with the precision[7] and recall[8] associated to each type of place.

[6]The confusion matrices always have five columns because, at each instant, the method can predict any of the five possible types of places. However, the number of rows depends on the actual occurrences of types of places in each scenario.
[7]Precision is the fraction of total predictions from some class (TP+FP) that really occurred (TP).
[8]Recall is the fraction of total occurrences from some class (TP+FN) that are properly predicted (TP).

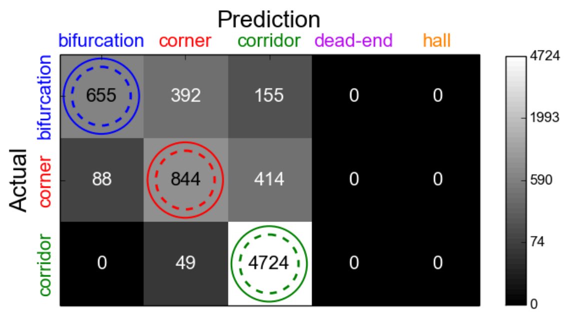
Figure 6.9: Classification results for simulated scenarios. **left**: confusion matrix showing predicted × actual values, with highlights of the most predicted value among the true occurrences of each type of region (solid circle), and the actual value that most occur among the predicted occurrences (dashed circle). **right**: table of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN), along with precision and recall for each type of place.



(a) S1 - Confusion Matrix

Place	TP	FP	FN	TN	Prec.	Rec.
bifurcation	7008	20	0	20662	99.7	100.0
corner	3674	32	120	23864	99.1	96.8
corridor	16856	100	32	10702	99.4	99.8
dead-end	0	0	0	27690	—	—
hall	0	0	0	27690	—	—

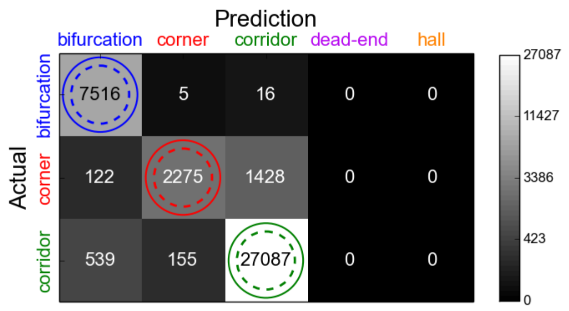
(b) S1 - Precision and Recall



(c) S2 - Confusion Matrix

Place	TP	FP	FN	TN	Prec.	Rec.
bifurcation	655	88	547	6031	88.2	54.5
corner	844	441	502	5534	65.7	62.7
corridor	4724	569	49	1979	89.2	99.0
dead-end	0	0	0	7321	—	—
hall	0	0	0	7321	—	—

(d) S2 - Precision and Recall



(e) S3 - Confusion Matrix

Place	TP	FP	FN	TN	Prec.	Rec.
bifurcation	7516	661	21	30945	91.9	99.7
corner	2275	160	1550	35158	93.4	59.5
corridor	27087	1444	694	9918	94.9	97.5
dead-end	0	0	0	39143	—	—
hall	0	0	0	39143	—	—

(f) S3 - Precision and Recall

Regarding the evaluation process, we count hits/misses in terms of observations associated to the regions, which means the prediction of a region has weight proportional to the length of the region. Additionally, it is hard to define an exact transition between two regions, such as corridors and bifurcations, therefore we allow small shifts in the limits of each word by applying a tolerance of $1m$ (10 observations) in the transitions between regions. Considering that each pixel in the ground-truth maps presented in Figures 6.1, 6.2 and 6.3 corresponds to a $0.1m \times 0.1m$ grid cell, we are allowing a variation of 10 pixels over the drawn borders between regions.

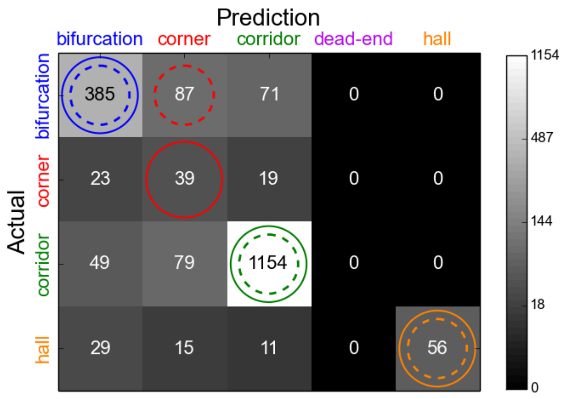
Analyzing the results obtained in simulated scenarios shown in Figure 6.9, we can see that the best results are, clearly, obtained in scenario S1 and the worst in scenario S3. Among the predicted occurrences of each type of place, the values that most occur are the expected ones, as shown by the dashed circles in the confusion matrix. Furthermore, the number of true positives in each class is always higher than the sum of false positives, which means precision above 50%. In fact, the average precisions are 99.4% in scenario S1, 81.0% in scenario S2, and 93.4% in scenario S3. The worst precision result (65.7%) happened with corners in scenario S2, due to some confusion between corners and bifurcations. The most predicted values among the actual occurrences of each type of place are also the expected ones, as shown by the solid circles in the confusion matrix. Likewise precision, the number of true positives is higher than the sum of false negatives in each class, thus all recalls stay above 50%. The average recalls are 98.9% in scenario S1, 72.1% in scenario S2, and 85.6% in scenario S3. The worst recalls occur with bifurcations (54.5%) and corners (62.7%) in scenario S2, and corners (59.5%) in scenario S3.

Concerning the issue with bifurcations, our method generated around zero false negatives in scenarios S1 and S3, nonetheless 45% of the occurrences of bifurcations in scenario S2 are false negatives. It seems, looking back at the map of Figure 6.1(d), that the number of free-space frontiers is not a good metric to classify bifurcations such as the ones in scenario S2 (actually, Figure 6.7 shows that the variation in the number of frontiers in bifurcations of scenario S2 is smaller than in other scenarios). Regarding corners, the problem is the confusion with corridors. Since corners are connected to corridors in all environments, we think that the main issue is probably the decision of where corridors end and corners start (and vice-versa).

Analyzing the results obtained in real scenarios shown in Figure 6.10, we observe a general decrease of quality in relation to the simulated scenarios. Notwithstanding, the quality of prediction drastically varies depending on the type of the place. The prediction of corridors and bifurcations is good: the average precision and average recall are respectively 86.4% and 89.5% for corridors, and 77.9% and 72.0% for bifurcations. Halls and dead-ends (or rooms) – the two types of places absent in simulated scenarios – are predicted with maximum precision, that is, there are no false positives of both types of places. However, the prediction of halls and dead-ends, although correct, only happens after the robot totally enters those two types of places, which implies in a moderate recall.

The worst performance in real scenarios is associated to corners, where the average precision and recall are only 27.4% and 49.7%, respectively. The results are specially

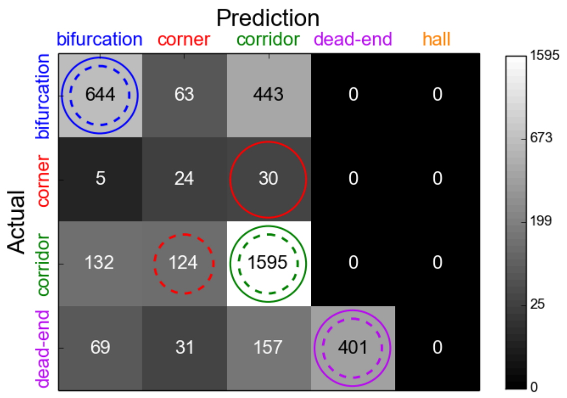
Figure 6.10: Classification results for real scenarios (see caption of Figure 6.9 for detailed description).



(a) R1 - Confusion Matrix

Place	TP	FP	FN	TN	Prec.	Rec.
bifurcation	385	101	158	1373	79.2	70.9
corner	39	181	42	1755	17.7	48.1
corridor	1154	101	128	634	92.0	90.0
dead-end	0	0	0	2017	—	—
hall	56	0	55	1906	100.0	50.5

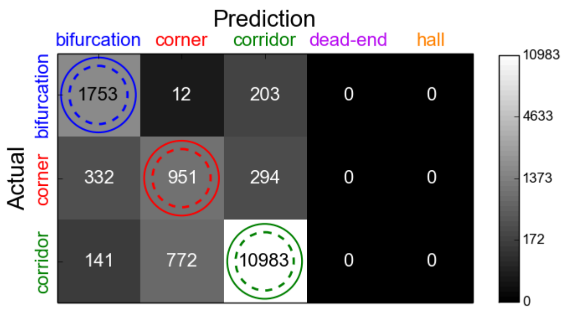
(b) R1 - Precision and Recall



(c) R2 - Confusion Matrix

Place	TP	FP	FN	TN	Prec.	Rec.
bifurcation	644	206	506	2362	75.8	56.0
corner	24	218	35	3441	9.9	40.7
corridor	1595	630	256	1237	71.7	86.2
dead-end	401	0	257	3060	100.0	60.9
hall	0	0	0	3718	—	—

(d) R2 - Precision and Recall



(e) R3 - Confusion Matrix

Place	TP	FP	FN	TN	Prec.	Rec.
bifurcation	1753	473	215	13000	78.8	89.1
corner	951	784	626	13080	54.8	60.3
corridor	10983	497	913	3048	95.7	92.3
dead-end	0	0	0	15441	—	—
hall	0	0	0	15441	—	—

(f) R3 - Precision and Recall

poor in scenarios R1 and R2. As shown in Figures 6.10(a) and (c), the actual values that most occur in the prediction of corners are bifurcations in scenario R1, and corridors in scenario R2. The percentage of false negatives is also high, particularly in scenario R2, where corridors are the most predicted place at actual occurrences of corners. The apparent problem with scenario R1, as shown in the map of Figure 6.2(a), is that there are some bifurcations that look very similar to corners (i.e. bifurcations with narrow passageways where the robot turned 90°) and only two actual corners, both with several open doors, that looked similar to bifurcations. In scenario R2, as shown in the map of

Figure 6.2(c), there is only one small corner, which is visited a single time (corresponding to 1.5% of all observations). This corner is connected to a long corridor at one side and to a large bifurcation at the other side, which makes it quite different from the corners in the other scenarios. Unfortunately, when considering the very few actual occurrences of corners, the impact of one prediction error is huge, which causes the high precision drop.

6.5 Summary

This chapter presents an in-depth study of the characteristics of words based on free-space density, focusing on semantic mapping. We analyzed the discernibility and consistency of words associated to five different types of places: bifurcation, corners, corridors, dead-ends (or rooms) and halls. While the experiments in previous chapters were made in highly structured and well-behaved scenarios, now, the evaluation process considered some real datasets from SLAM literature with high presence of dynamic obstacles, apertures (e.g open doors) and a few unusual structures.

The analysis of the word components (i.e. the variation of free-space density in each type of place, region length and variation in the robot orientation) showed that corridors, dead-ends and halls are easy to distinguish. On the other hand, bifurcations and corners are usually very similar, thus it is very hard to tell the difference between them. As expected, this is a serious issue for semantic classification, however it does not strongly affect our place recognition strategy. In such problem we only search for matches among the existing words, it does not matter to what places the words are associated, and, more importantly, we consider long sequences of words to reduce ambiguities.

As a matter of fact, we intend to investigate the use of sequential information as a way to improve the process of semantic classification. For instance, we may use n-grams to predict what place comes after some sequence of n places (e.g. corridor-$\cdots$-corner-corridor). Still, this type of strategy seems to be heavily map dependent, which, although good for place recognition, is not suitable for semantic classification. Actually, sequences of places that are common in some environments may not appear at all in other environments.

Nevertheless, considering that the original information of the three syllables from our proposed words did not seem enough to provide an easy semantic classification of all places, we studied another type of kernel-based information, which is the number of frontiers of free-space in the surroundings of the robot. This metric proved to be good to distinguish bifurcations from corners, however dynamic obstacles and apertures in the environment (that usually happen in corridors) eventually generate unexpected results.

Lastly, a simple strategy for semantic classification of the words is presented in the form of a manually-defined decision tree, which is based on ideas captured in the analysis. Good results were obtained in the simulated scenarios, but some problems have arisen in real scenarios, in special concerning the classification of corners. The problems mainly originate from the high number of unexpected apertures in the environments, the high dynamism of the scenarios (i.e. too many people passing by the robot), and the unpredictability of the robot trajectory. In spite of that, this basic strategy achieved precisions, in real scenarios, of 86.4% and 77.9% for corridors and bifurcations, and 100% for halls and dead-ends.

We have shown in this chapter that the addition of semantic information to the words describing the regions visited by the robot is a feasible task. Moreover, considering the acquired results, the achievable lower bound of the classification process is good for several types of places. As shown by many methods of semantic mapping (KOSTAVELIS; GASTERATOS, 2015; MOZOS, 2010; PRONOBIS; JENSFELT, 2012), the application of techniques from machine learning can be useful to improve the quality of the classification, and therefore is a valuable starting point for future work.

7 CONCLUSION AND FUTURE WORK

In this thesis, we have studied a way to view the problem of state estimation in robotics through the lens of linguistic processing. Our focus is on the place recognition problem, which is a fundamental task for localization and mapping, and consists in determining if different observations made by a robot are taken from the same place. Most methods in literature usually compare raw sensor readings or extract salient features in the environment and compare them. The problem is that practically all environments are highly ambiguous, specially structured ones, leading to wrong associations.Our approach translates long sequences of raw measurements into an efficient, compact, and much more abstract, text representation.

Summing up, we propose a novel observation model, called Free-Space Density (FSD), based on kernel density estimates of the known free-space in a region. The model is robust to moderate variations in robot position, and, due to its orientation independence, can represent each position of the environment with a single value. It also implies in low memory consumption and exceptional performance in comparisons (e.g. comparing pairs of observations is equivalent to a single floating-point operation). FSD was successfully applied to the global localization problem of a mobile robot using particle filters. Experiments showed similar quality of results, and even better quality in some cases than other traditional techniques, however with much smaller computational time. Basically, the efficiency of our novel observation model allowed an impressive increase in the number of particles, which was good for compensating the high dimensionality reduction of sensors readings.

Given that density transitions are smooth, the quantization of FSD into classes of densities allows the creation of regions of homogeneous density that can be represented in a very compact way – words. The word representation of a region is composed of three syllables describing the general free-space density, the size and the orientation variation of that region. With such compact representation we are able to easily classify topological and metric information of the environment. After converting raw sensor readings into words, all observations made by the robot are represented in a simple text. Place recognition is performed by searching for matches of words in texts. However, ambiguous environments lead to ambiguities in the text, which are solved by analyzing sequences of words (i.e. n-grams). Our place recognition method simply searches for matches of n-grams. The larger the length of the n-gram, the more probable of being correct is the obtained match.

Despite being originally aimed for static environments, our approach can also be extended to long-term mobile robot operation, where we must deal with not only static or highly dynamic objects, but also semi-static objects, which can move occasionally (e.g. doors, furniture, etc). If a word describing some region is not associated to a static object, then we must consider the possibility of not finding this particular word when revisiting this region. The core of our long-term place recognition proposal is to generate alternative "corrected" words for all regions. Every time a word is built, we also consider the case where the density change was brought by a non-static object. In this case, we create a new word sequence assuming the previous density remained present in the new word's location. Experiments made in real and simulated scenarios have shown that this extension of our word-based approach produced improvements in the matching results.

Finally, we perform an analysis of the proposed FSD-based words in diverse types of simulated and real scenarios. We evaluate if words associated to specific types of places, such as corridors, bifurcations and corners, have noticeable features that may be used to distinguish them, and thus, to serve as basis for an approach of semantic mapping. For each word built, we check information such as the variation of free-space density, the variation in the robot orientation, the region length associated to the word, and the variation in the number of free-space frontiers. A simple classification strategy is proposed based on some notions obtained during the analysis and the performance was generally good. The biggest issues occurred in real scenarios due to distortions in the free-space density caused by unexpected apertures that lead to confusion between corners/corridors and bifurcations; or places with high presence of dynamic obstacles.

Even though the techniques presented in this thesis have obtained good results that support our proposal, there are numerous aspects that can be improved in further studies. In the future, we want to investigate the use of other information sources, besides free-space density, to build words. In Section 6.3, we verify that the variation in the number of frontiers of free-space surrounding the robot can be used to improve disambiguation among different types of places. Likewise, many other features can be explored, like the shape of the regions where the robot is located, which can be contemplated using metrics such as graph spectrum (OLIVEIRA; SILVA; BARONE, 2015) or Voronoi skeletonization (SAEEDI et al., 2012). Furthermore, we could try to explore information from other types of sensors (e.g. cameras), such as the distribution of visual features in the environment, or even the density of colors in images.

A drawback of our place recognition strategy is that we only search for sequential matches in the direction visited by the robot, which means that if the robot traverses a region twice, but with opposite directions, we will probably not detect the true correspondences. The problem is that, since the robot field-of-view is limited (180°), the observations made by the robot can be far different depending on what direction the robot is heading. As a result, the generated words tend to be slightly different. A possible form of minimizing such problem is to try to predict the observations behind the robot, and also to study ways of generating reverse words.

One of the main limitations of our approach is that it relies on a handful of empirically chosen fixed parameters. Some of them are dependent of the environment, for instance, the kernel size used to compute the free-space density cannot be too small or too big, otherwise the density will almost not vary. As we observed in experiments of Chapter 3, the ideal kernel size is the usual width of corridors in the environment (around $3 \sim 4$ meters), which makes our method suitable for indoor environments. Other parameters, like the minimum n-gram size required for accepting matches depends on the characteristics of the environment and on the robot trajectory. In highly ambiguous scenarios we only must allow matches of very long sequences of words. Due to this, the strategy to determine such threshold is still an open issue. A possible, but burdensome, approach to circumvent this problem of parameters tuning is to study the migration of our proposal to a probabilistic paradigm, that incorporates noise and uncertainty. For example, instead of words we would deal with states, and instead of n-grams we would deal with hidden Markov models.

Another area of study with enormous potential is word classification and semantic mapping. Although the latest contribution of this thesis is a simple classification strategy with a decision-tree that generally works well, there is plenty of room for improvements with the study of machine learning techniques. Some of the most promising techniques that are widely used in natural language processing are Word Embeddings (BENGIO et al., 2003). Those techniques perform the mapping of words (and sequences of words) to vectors of numbers, which can be made using neural networks, probabilistic models (e.g. n-grams models), among others. The main idea is that words sharing semantical similarities should be represented near each other in the vectorial space. Applying word embeddings in our approach, we could automatically learn the best parameters to describe corridors, bifurcations, etc. Additionaly, we could use this type of approach to extend our proposal to handle, not only the information acquired by laser range-finders, but also the information captured by cameras.

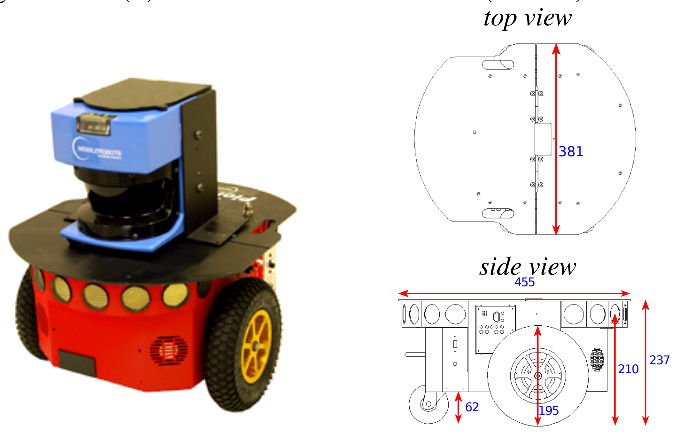
Finally, we can apply some of our techniques in other contexts such as the motion planning problem. For instance, strategies of integrated exploration (JORGE et al., 2015; MAFFEI et al., 2014) and robot path planning (MACHARET et al., 2013; PRESTES; IDIART, 2010) may use the proposed textual representation of the robot path as a high-level guide for the planning algorithms.

Appendices

AppendixA EXPERIMENT APPARATUS - P3-DX MOBILE ROBOT

All techniques proposed in this work were designed and evaluated using the Pioneer 3-DX mobile robot from *Adept MobileRobots*, which is a small lightweight two-wheel two-motor differential drive robot for indoor use, shown in Figure A.1. The robot is equipped with a SICK LMS-200 laser range finder, which is able to scan $180°$, with maximum resolution of $0.25°$ and maximum range of $80m$. A detailed description of the robot specifications is shown in Table A.1.

Figure A.1: Pioneer 3-DX mobile robot. (a) Picture of the robot equipped with a SICK LMS-200 laser range finder. (b) Dimensions of the robot (in mm).

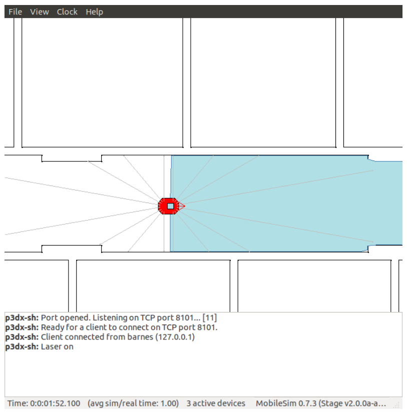


(a) (b)

Besides experiments with the real robot, we also performed simulations using the MobileSim software, which is an open source 2D simulator from *Adept MobileRobots*. Figure A.2 shows a screenshot of the simulator.

All experiments were performed in a notebook with a 4x Intel ® Core ™ i7 processor with 16GB of RAM memory.

Figure A.2: Screenshot of the MobileSim simulator. The robot is shown in red, laser readings are shown in blue, obstacles are shown in black.



```
p3dx-sh: Port opened. Listening on TCP port 8101... [11]
p3dx-sh: Ready for a client to connect on TCP port 8101.
p3dx-sh: Client connected from barnes (127.0.0.1)
p3dx-sh: Laser on
```

```
Time: 0:0:01:52.100   (avg sim/real time: 1.00)   3 active devices   MobileSim 0.7.3 (Stage v2.0.0a-a...
```

Table A.1: Specifications of the robot.

Physical Characteristics	
Length (cm)	45.5
Width (cm)	38.1
Height (cm)	23.7
Clearance (cm)	6.0
Clearance bumpers (cm)	3.5
Weight (kg)	9
Payload (kg)	25
Mobility	
Wheels	2 foam-filled
tread	knobby
diam (mm)	195.3
width (mm)	47.4
Caster(mm)	75
Steering	Differential
Gear ratio	38.3:1
Swing (cm)	26.7
Turn (cm)	0
Translate speed max (mm/sec)	1,400
Rotate speed max (deg/sec)	300
Traversable step max (mm)	20
Traversable gap max (mm)	89
Traversable slope max (grade)	25%
Traversable terrains	Wheel-chair accessible
Sensors	
SICK LMS-200 laser range finder	
Angular resolution (deg)	0.25; 0.5; 1
Scanning angle (deg)	180
Resolution (mm)	10
Range max (m)	80
Sonar Front Array (one each side, six forward @ 20 intervals)	8
Encoders (2 ea)	
counts/rev	76,600
counts/mm	128
counts/rotation	33,500

AppendixB DERIVATION OF THE GAUSS-NEWTON ALGORITHM FOR GRAPH-BASED SLAM

Following the concepts introduced in Section 2.1.3 of this thesis and the work described in (GRISETTI et al., 2010b), we present the derivation of the Gauss-Newton algorithm used for least squares optimization of graph-based SLAM. Its goal is to find the optimal configuration, $\boldsymbol{s}^*$, which minimizes the sum of squared errors of all edges, given the current state vector $\boldsymbol{s}$.

$$F(\boldsymbol{s}) = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})^T \, \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \, \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s}), \tag{B.1}$$

$$\boldsymbol{s}^* = \operatorname*{argmin}_{\boldsymbol{s}} F(\boldsymbol{s}). \tag{B.2}$$

The first step to solve Equation B.2 is linearizing the error around a guess $\breve{\mathrm{s}}$, through a first-order Taylor series expansion,

$$\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) \simeq \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{s} \tag{B.3}$$

where $\boldsymbol{J}_{ij}$ is the Jacobian of the error computed at $\breve{\mathrm{s}}$, which is non-zero only in the blocks associated to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and it has a $3 \times 3n$ dimension (i.e. $\boldsymbol{J}_{ij}$ has a 3×3 block of partial derivatives for each one of the n nodes):

$$\begin{aligned}
\boldsymbol{J}_{ij} &= \left.\frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{s}}\right|_{\boldsymbol{s}=\breve{\boldsymbol{s}}} \\
&= \left(\frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{x}_0} \cdots \frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{x}_i} \cdots \frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{x}_j} \cdots \frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})}{\partial \boldsymbol{x}_t} \right) \\
&= \left(\boldsymbol{0} \cdots \boldsymbol{0} \ \underbrace{\boldsymbol{A}_{ij}}_{\frac{\partial \boldsymbol{e}_{ij}(\boldsymbol{s})}{\partial \boldsymbol{x}_i}} \ \boldsymbol{0} \cdots \boldsymbol{0} \ \underbrace{\boldsymbol{B}_{ij}}_{\frac{\partial \boldsymbol{e}_{ij}(\boldsymbol{s})}{\partial \boldsymbol{x}_j}} \ \boldsymbol{0} \cdots \boldsymbol{0} \right)
\end{aligned} \tag{B.4}$$

We approximate $F(\boldsymbol{s})$ with $F(\breve{\boldsymbol{s}} + \Delta\boldsymbol{s})$ by applying the results of Equation B.3 in

Equation B.1:

$$F(\breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \Big(\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}} + \Delta\boldsymbol{s})^T \, \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \, \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) \Big) \tag{B.5}$$

$$\simeq \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \Big(\big(\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{s} \big)^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \big(\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) + \boldsymbol{J}_{ij}\Delta\boldsymbol{s} \big) \Big) \tag{B.6}$$

$$= \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \Big(\underbrace{\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})}_{T_1} + \underbrace{\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{J}_{ij}\Delta\boldsymbol{s}}_{T_2}$$

$$+ \underbrace{\Delta\boldsymbol{s}^T \boldsymbol{J}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})}_{T_3} + \underbrace{\Delta\boldsymbol{s}^T \boldsymbol{J}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{J}_{ij}\Delta\boldsymbol{s}}_{T_4} \Big) \tag{B.7}$$

Now, we analyze each term of Equation B.7. The first term, T_1, corresponds to a scalar,

$$T_1 = \underbrace{\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})^T}_{1\times 3} \underbrace{\boldsymbol{\Omega}(\boldsymbol{a}_{ij})}_{3\times 3} \underbrace{\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})}_{3\times 1} = c_{ij}. \tag{B.8}$$

The sum of the second and third terms, $(T_2 + T_3)$, can be redefined following a basic property of matrices manipulation[1]. Then, we rewrite the resulting term, by isolating $\Delta\boldsymbol{s}$,

$$T_2 + T_3 = 2T_2 = 2 \underbrace{\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})^T}_{1\times 3} \underbrace{\boldsymbol{\Omega}(\boldsymbol{a}_{ij})}_{3\times 3} \underbrace{\boldsymbol{J}_{ij}}_{3\times 3n} \Delta\boldsymbol{s} = 2 \underbrace{\boldsymbol{b}_{ij}^T}_{1\times 3n} \Delta\boldsymbol{s}. \tag{B.9}$$

Finally, the last term, T_4, is also rewritten by isolating $\Delta\boldsymbol{s}$,

$$T_4 = \Delta\boldsymbol{s}^T \underbrace{\boldsymbol{J}_{ij}^T}_{3n\times 3} \underbrace{\boldsymbol{\Omega}(\boldsymbol{a}_{ij})}_{3\times 3} \underbrace{\boldsymbol{J}_{ij}}_{3\times 3n} \Delta\boldsymbol{s} = \Delta\boldsymbol{s}^T \underbrace{\boldsymbol{H}_{ij}}_{3n\times 3n} \Delta\boldsymbol{s} \tag{B.10}$$

We replace the new terms in Equation B.7:

$$F(\breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) \simeq \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \Big(c_{ij} + 2\boldsymbol{b}_{ij}^T \Delta\boldsymbol{s} + \Delta\boldsymbol{s}^T \boldsymbol{H}_{ij} \Delta\boldsymbol{s} \Big) \tag{B.11}$$

Applying the summation, we obtain

$$F(\breve{\boldsymbol{s}} + \Delta\boldsymbol{s}) \simeq c + 2\boldsymbol{b}^T \Delta\boldsymbol{s} + \Delta\boldsymbol{s}^T \boldsymbol{H} \Delta\boldsymbol{s} \tag{B.12}$$

[1]Following equation 5 of section 1 from the Matrix Cookbook (PETERSEN; PEDERSEN, 2012), the transposition of the product of matrices, $(\boldsymbol{ABCD})^T$, is equal to the reversed product of transposed matrices, $\boldsymbol{D}^T\boldsymbol{C}^T\boldsymbol{B}^T\boldsymbol{A}^T$. Thus, $\boldsymbol{ABCD} + \boldsymbol{D}^T\boldsymbol{C}^T\boldsymbol{B}^T\boldsymbol{A}^T = \boldsymbol{ABCD} + (\boldsymbol{ABCD})^T$, which is equal to $2\boldsymbol{ABCD}$. This can be applied in our case because $\boldsymbol{\Omega}(\boldsymbol{a}_{ij})$ is symmetric, i.e. $\boldsymbol{\Omega}(\boldsymbol{a}_{ij}) = \boldsymbol{\Omega}(\boldsymbol{a}_{ij})^T$.

where

$$c = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} c_{ij} \quad \text{and} \quad \boldsymbol{b}^T = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \boldsymbol{b}_{ij}^T \quad \text{and} \quad \boldsymbol{H} = \sum_{\boldsymbol{a}_{ij} \in \mathcal{A}} \boldsymbol{H}_{ij}.$$

The matrix $\boldsymbol{H}$ is the information matrix of the full state, obtained using the Jacobians to project the measurement error into the trajectories space. $\boldsymbol{H}$ is sparse because the Jacobians that compose it are also sparse. In fact, each constraint $\boldsymbol{a}_{ij}$ between two nodes only updates two non-zeroes blocks in the vector $\boldsymbol{b}_{ij}$ and four non-zeroes blocks in the matrix $\boldsymbol{H}_{ij}$ (they are previously defined in Equations B.9 and B.10, respectively), as shown next,[2]

$$\begin{aligned} \boldsymbol{b}_{ij} &= \left(\boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}})^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{J}_{ij} \right)^T = \boldsymbol{J}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) \\ &= \begin{pmatrix} \vdots \\ \boldsymbol{A}_{ij}^T \\ \vdots \\ \boldsymbol{B}_{ij}^T \\ \vdots \end{pmatrix} \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) = \begin{pmatrix} \vdots \\ \boldsymbol{A}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) \\ \vdots \\ \boldsymbol{B}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{e}(\boldsymbol{a}_{ij}, \breve{\boldsymbol{s}}) \\ \vdots \end{pmatrix} \end{aligned} \tag{B.13}$$

$$\begin{aligned} \boldsymbol{H}_{ij} &= \boldsymbol{J}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{J}_{ij} \\ &= \begin{pmatrix} \vdots \\ \boldsymbol{A}_{ij}^T \\ \vdots \\ \boldsymbol{B}_{ij}^T \\ \vdots \end{pmatrix} \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \begin{pmatrix} \cdots & \boldsymbol{A}_{ij} & \cdots & \boldsymbol{B}_{ij} & \cdots \end{pmatrix} \\ &= \begin{pmatrix} \ddots & & & & \\ & \boldsymbol{A}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{A}_{ij} & \cdots & \boldsymbol{A}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{B}_{ij} & \\ & \vdots & \ddots & \vdots & \\ & \boldsymbol{B}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{A}_{ij} & \cdots & \boldsymbol{B}_{ij}^T \boldsymbol{\Omega}(\boldsymbol{a}_{ij}) \boldsymbol{B}_{ij} & \\ & & & & \ddots \end{pmatrix} \end{aligned} \tag{B.14}$$

We can find the optimal $\Delta \boldsymbol{s}^*$ that minimizes the approximated $F(\breve{\boldsymbol{s}} + \Delta \boldsymbol{s})$ by comput-

[2]We omitted the zeroes blocks for simplicity of notation.

ing the derivative[3] of Equation B.12 and setting it to zero.

$$\frac{\partial F(\breve{\boldsymbol{s}} + \Delta \boldsymbol{s})}{\partial \Delta \boldsymbol{s}} \simeq 2\boldsymbol{b} + 2\boldsymbol{H}\Delta \boldsymbol{s}^* = 0 \tag{B.15}$$

The solution for the increment can be obtained by solving the linear system

$$\boldsymbol{H}\Delta \boldsymbol{s}^* = -\boldsymbol{b}. \tag{B.16}$$

Then, we obtain the resulting state by adding $\Delta \boldsymbol{s}^*$ to the initial guess.

$$\boldsymbol{s}^* = \breve{\boldsymbol{s}} + \Delta \boldsymbol{s}^* \tag{B.17}$$

To conclude, we present the detailed values of the error function, $\boldsymbol{e}(\boldsymbol{a}_{ij}, \boldsymbol{s})$, and the Jacobians of the error, $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$, which must be determined for each constraint added to the graph. After computing these three fundamental components, we can trivially build the matrix $\boldsymbol{H}$ and vector $\boldsymbol{b}$ through matrices multiplication, and proceed to solve the linear system of Equation 2.11.

Just for text simplification, we define the following notation shown in Table B.1 to refer to components of $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ and $\boldsymbol{r}(\boldsymbol{a}_{ij})$.

Table B.1: Simplified notation used in this section

$\boldsymbol{x}_i = (x_i, y_i, \theta_i)$	$s_i = \sin\theta_i$	$c_i = \cos\theta_i$
$\boldsymbol{x}_j = (x_j, y_j, \theta_j)$		
$\boldsymbol{r}(\boldsymbol{a}_{ij}) = (x_{ij}, y_{ij}, \theta_{ij})$	$s_{ij} = \sin\theta_{ij}$	$c_{ij} = \cos\theta_{ij}$

[3]Following equations 69 and 81 of section 2.4 from the Matrix Cookbook (PETERSEN; PEDERSEN, 2012), the derivative of a quadratic form function $f(\boldsymbol{x}) = \boldsymbol{b}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$ is equal to $\frac{\partial f(\boldsymbol{x})}{\boldsymbol{x}} = \boldsymbol{b} + (\boldsymbol{H} + \boldsymbol{H}^T)\boldsymbol{x}$. Since in our case $\boldsymbol{H}$ is symmetric, then $(\boldsymbol{H} + \boldsymbol{H}^T) = 2\boldsymbol{H}$.

$$\begin{aligned}
\boldsymbol{e}(\boldsymbol{a}_{ij},\boldsymbol{s}) = \hat{\boldsymbol{r}}(\boldsymbol{a}_{ij},\boldsymbol{s}) \ominus \boldsymbol{r}(\boldsymbol{a}_{ij}) &= (\boldsymbol{x}_j \ominus \boldsymbol{x}_i) \ominus \boldsymbol{r}(\boldsymbol{a}_{ij}) \\
&= \begin{pmatrix} c_{ij} & s_{ij} & 0 \\ -s_{ij} & c_{ij} & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(\begin{pmatrix} c_i & s_i & 0 \\ -s_i & c_i & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_j - x_i \\ y_j - y_i \\ \theta_j - \theta_i \end{pmatrix} - \begin{pmatrix} x_{ij} \\ y_{ij} \\ \theta_{ij} \end{pmatrix} \right) \\
&= \begin{pmatrix} c_{ij} & s_{ij} & 0 \\ -s_{ij} & c_{ij} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_i(x_j - x_i) + s_i(y_j - y_i) - x_{ij} \\ -s_i(x_j - x_i) + c_i(y_j - y_i) - y_{ij} \\ \theta_j - \theta_i - \theta_{ij} \end{pmatrix} \\
&= \begin{pmatrix} (c_{ij}c_i - s_{ij}s_i)(x_j - x_i) + (c_{ij}s_i + s_{ij}c_i)(y_j - y_i) - c_{ij}x_{ij} - s_{ij}y_{ij} \\ (-s_{ij}c_i - c_{ij}s_i)(x_j - x_i) + (-s_{ij}s_i + c_{ij}c_i)(y_j - y_i) + s_{ij}x_{ij} - c_{ij}y_{ij} \\ \theta_j - \theta_i - \theta_{ij} \end{pmatrix}
\end{aligned} \tag{B.18}$$

$$\begin{aligned}
\boldsymbol{A}_{ij} &= \frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij},\boldsymbol{s})}{\partial \boldsymbol{x}_i} \\
&= \begin{pmatrix} (-c_{ij}c_i + s_{ij}s_i) & (-c_{ij}s_i - s_{ij}c_i) & (-c_{ij}s_i - s_{ij}c_i)(x_j - x_i) + (c_{ij}c_i - s_{ij}s_i)(y_j - y_i) \\ (s_{ij}c_i + c_{ij}s_i) & (s_{ij}s_i - c_{ij}c_i) & (s_{ij}s_i - c_{ij}c_i)(x_j - x_i) + (-s_{ij}c_i - c_{ij}s_i)(y_j - y_i) \\ 0 & 0 & -1 \end{pmatrix}
\end{aligned} \tag{B.19}$$

$$\boldsymbol{B}_{ij} = \frac{\partial \boldsymbol{e}(\boldsymbol{a}_{ij},\boldsymbol{s})}{\partial \boldsymbol{x}_j} = \begin{pmatrix} (c_{ij}c_i - s_{ij}s_i) & (c_{ij}s_i + s_{ij}c_i) & 0 \\ (-s_{ij}c_i - c_{ij}s_i) & (-s_{ij}s_i + c_{ij}c_i) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.20}$$

AppendixC KERNEL DENSITY ESTIMATION

In statistics, a probability density function f is a natural description of the distribution of a random variable X. The probability of X falling within a specific range of values a and b, is given by the integral of f over the range (a, b). The probability density function is always non-negative and adds to one over the entire space (SILVERMAN, 1986; SCOTT, 1992).

$$p(a < X < b) = \int_a^b f(x)\,dx \qquad \forall a < b \tag{C.1}$$

$$p(a < X < b) = \int_{-\infty}^{\infty} f(x)\,dx = 1 \tag{C.2}$$

Intuitively, the probability density, $f(x)$, at a given point x can be expressed as

$$f(x) = \lim_{h \to 0} \frac{p(x - h < X < x + h)}{2h}, \tag{C.3}$$

where $p(x - h < X < x + h)$ is the probability of sampling X inside the range $(x - h, x + h)$, and $2h$ is the width of the range.

In practice, we have

$$\hat{f}(x) = \frac{1}{2h}\Big(\frac{n_x}{n}\Big), \tag{C.4}$$

where n_x is the number of samples that fall inside the range $(x - h, x + h)$, from a total of n samples.

Defining a kernel function K_h as

$$K_h(d) = \begin{cases} \dfrac{1}{2h} & \text{if } |d| \leq h \\ 0 & \text{otherwise} \end{cases}, \tag{C.5}$$

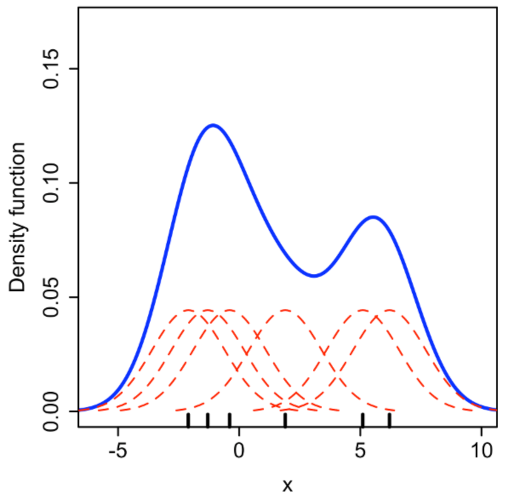
we can rewrite Equation C.4 as the kernel density estimation (KDE) given by

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i), \tag{C.6}$$

where x_i is the $i-$th sample of X.

However, instead of using the simple uniform kernel profile, described in Equation C.5, many other different profiles can be used to estimate a pdf. Provided the kernel itself is a probability density function and non-negative everywhere, the resulting KDE will be a proper probability density function. Additionaly, the KDE will inherit the continuity and differentiability properties of the kernel (SILVERMAN, 1986). Figure C.1 shows a KDE of an arbitrary function built from 6 samples using a Gaussian kernel profile.

Figure C.1: Kernel density estimate (blue) obtained from the application of individual Gaussian kernels (red) over 6 samples (black). Figure extracted from Wikipedia.[1]



[1]Available in <https://en.wikipedia.org/wiki/Kernel_density_estimation>

REFERENCES

AGARWAL, P. **Robust Graph-Based Localization and Mapping**. Thesis (PhD) — University of Freiburg, Department of Computer Science, April 2015.

ALTHAUS, P.; CHRISTENSEN, H. I. Behavior coordination in structured environments. **Advanced Robotics**, v. 17, n. 7, p. 657–674, 2003. Available from Internet: <http://dx.doi.org/10.1163/156855303769157009>.

ARNDT, M.; WILLE, S.; SOUZA, L. d.; REY, V. F.; WEHN, N.; BERNS, K. Performance evaluation of ambient services by combining robotic frameworks and a smart environment platform. **Robotics and Autonomous Systems**, v. 61, n. 11, p. 1173–1185, 2013. ISSN 0921-8890. Ubiquitous Robotics. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0921889013000614>.

BACKUS, J. W. The syntax and semantics of the proposed international algebraic language of the zurich acm-gamm conference. In: **Proceedings of the International Conference on Information Processing**. Paris, France: UNESCO, 1959. p. 125–131.

BARRÓN-CEDEÑO, A.; ROSSO, P. On automatic plagiarism detection based on n-grams comparison. In: BOUGHANEM, M.; BERRUT, C.; MOTHE, J.; SOULE-DUPUY, C. (Ed.). **Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval**. Berlin, Heidelberg: Springer-Verlag, 2009. (ECIR '09), p. 696–700. ISBN 978-3-642-00958-7.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: LEONARDIS, A.; BISCHOF, H.; PINZ, A. (Ed.). **Computer Vision – ECCV 2006**. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2006. v. 3951, chp. Lecture Notes in Computer Science, p. 404–417. ISBN 978-3-540-33832-1.

BEESON, P.; JONG, N. K.; KUIPERS, B. Towards autonomous topological place detection using the extended voronoi graph. In: **Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2005. p. 4373–4379.

BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JANVIN, C. A neural probabilistic language model. **Journal of Machine Learning Research**, v. 3, p. 1137–1155, 2003. Available from Internet: <http://dblp.uni-trier.de/db/journals/jmlr/jmlr3.html#BengioDVJ03>.

BESL, P. J.; MCKAY, N. D. A method for registration of 3-d shapes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE Computer Society, Washington, DC, USA, v. 14, n. 2, p. 239–256, feb 1992. ISSN 0162-8828. Available from Internet: <http://dx.doi.org/10.1109/34.121791>.

BIBER, P.; DUCKETT, T. Dynamic maps for long-term operation of mobile service robots. In: **Proceedings of the 2005 Robotics: Science and Systems (RSS) Conference**. Cambridge, MA, USA: MIT Press, 2005.

BIBER, P.; DUCKETT, T. Experimental analysis of sample-based maps for long-term slam. **The International Journal of Robotics Research**, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 28, n. 1, p. 20–33, jan 2009. ISSN 0278-3649. Available from Internet: <http://dx.doi.org/10.1177/0278364908096286>.

BORENSTEIN, J.; KOREN, Y. Histogramic in-motion mapping for mobile robot obstacle avoidance. **IEEE Transactions on Robotics and Automation**, v. 7, n. 4, p. 535–539, Aug 1991. ISSN 1042-296X.

BOSSE, M.; NEWMAN, P.; LEONARD, J.; SOIKA, M.; FEITEN, W.; TELLER, S. An atlas framework for scalable mapping. In: **Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2003. v. 2, p. 1899–1906. ISSN 1050-4729.

BOSSE, M.; ZLOT, R. Map matching and data association for large-scale two-dimensional laser scan-based slam. **The International Journal of Robotics Research**, v. 27, n. 6, p. 667–691, 2008. Available from Internet: <http://ijr.sagepub.com/content/27/6/667.abstract>.

BURGARD, W.; CREMERS, A. B.; FOX, D.; HÄHNEL, D.; LAKEMEYER, G.; SCHULZ, D.; STEINER, W.; THRUN, S. The interactive museum tour-guide robot. In: **Proceedings of the 1998 National Conference on Artificial Intelligence (AAAI)**. Menlo Park, CA, USA: AAAI Press, 1998.

BURGARD, W.; DERR, A.; FOX, D.; CREMERS, A. B. Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach. In: **Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 1998. v. 2, p. 730–735.

BURGARD, W.; HEBERT, M. World modeling. In: **Springer handbook of robotics**. Berlin, Heidelberg: Springer, 2008. p. 853–869.

CADENA, C.; CARLONE, L.; CARRILLO, H.; LATIF, Y.; SCARAMUZZA, D.; NEIRA, J.; REID, I.; LEONARD, J. J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. **IEEE Transactions on Robotics**, v. 32, n. 6, p. 1309–1332, Dec 2016. ISSN 1552-3098.

CHEN, C.; WANG, H. Appearance-based topological bayesian inference for loop-closing detection in a cross-country environment. **The International Journal of Robotics Research**, v. 25, n. 10, p. 953–983, 2006.

CHENG, Y. Mean shift, mode seeking, and clustering. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 17, n. 8, p. 790–799, Aug 1995. ISSN 0162-8828.

CHOI, Y.-H.; LEE, T.-K.; OH, S.-Y. A line feature based slam with low grade range sensors using geometric constraints and active exploration for mobile robot. **Autonomous Robots**, v. 24, n. 1, p. 13–27, 2008.

CHOMSKY, N. Three models for the description of language. **IRE Transactions on Information Theory**, v. 2, n. 3, p. 113–124, September 1956. ISSN 0096-1000.

CHOSET, H.; NAGATANI, K. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. **IEEE Transactions on Robotics and Automation**, v. 17, n. 2, p. 125–137, Apr 2001. ISSN 1042-296X.

COX, I. J. Blanche - an experiment in guidance and navigation of an autonomous robot vehicle. **IEEE Transactions on Robotics and Automation**, v. 7, n. 2, p. 193–204, Apr 1991. ISSN 1042-296X.

COX, I. J.; LEONARD, J. J. Modeling a dynamic environment using a bayesian multiple hypothesis approach. **Artificial Intelligence**, v. 66, n. 2, p. 311–344, 1994. ISSN 0004-3702. Available from Internet: <http://www.sciencedirect.com/science/article/pii/0004370294900299>.

CUMMINS, M.; NEWMAN, P. Probabilistic appearance based navigation and loop closing. In: **Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2007. p. 2042–2048. ISSN 1050-4729.

CUMMINS, M.; NEWMAN, P. Fab-map: Probabilistic localization and mapping in the space of appearance. **The International Journal of Robotics Research**, v. 27, n. 6, p. 647–665, 2008. Available from Internet: <http://ijr.sagepub.com/content/27/6/647.abstract>.

CUMMINS, M.; NEWMAN, P. Highly scalable appearance-only slam - fab-map 2.0. In: **Proceedings of the 2009 Robotics: Science and Systems (RSS) Conference**. Cambridge, MA, USA: MIT Press, 2009.

DELLAERT, F.; FOX, D.; BURGARD, W.; THRUN, S. Monte carlo localization for mobile robots. In: **Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 1999.

DELLAERT, F.; KAESS, M. Square root sam: Simultaneous localization and mapping via square root information smoothing. **The International Journal of Robotics Research**, v. 25, n. 12, p. 1181–1203, 2006. Available from Internet: <http://ijr.sagepub.com/content/25/12/1181.abstract>.

DEWEY, G. **Relative Frequency of English Speech Sounds**. Cambridge, MA, USA: Harvard University Press, 1923. (Harvard Studies in Education).

DUCKETT, T.; NEHMZOW, U. Mobile robot self-localisation using occupancy histograms and a mixture of gaussian location hypotheses. **Robotics and Autonomous Systems**, v. 34, n. 2–3, p. 117–129, 2001. ISSN 0921-8890. European Workshop on Advanced Mobile Robots.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. **Robotics Automation Magazine, IEEE**, v. 13, n. 2, p. 99–110, june 2006. ISSN 1070-9932.

ELGAMMAL, A.; DURAISWAMI, R.; HARWOOD, D.; DAVIS, L. S. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. **Proceedings of the IEEE**, v. 90, n. 7, p. 1151–1163, Jul 2002. ISSN 0018-9219.

ELIAZAR, A. I.; PARR, R. Dp-slam 2.0. In: **Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2004. v. 2, p. 1314–1320. ISSN 1050-4729.

ESTRADA, C.; NEIRA, J.; TARDOS, J. D. Hierarchical slam: Real-time accurate mapping of large environments. **IEEE Transactions on Robotics**, IEEE Press, Piscataway, NJ, USA, v. 21, n. 4, p. 588–596, aug. 2005. ISSN 1552-3098.

FOX, D.; BURGARD, W.; THRUN, S. Markov localization for mobile robots in dynamic environments. **Journal of Artificial Intelligence Research**, p. 391–427, 1999.

FRESE, U. Treemap: An o(log n) algorithm for indoor simultaneous localization and mapping. **Autonomous Robots**, Kluwer Academic Publishers, Hingham, MA, USA, v. 21, n. 2, p. 103–122, sep 2006. ISSN 0929-5593. Available from Internet: <http://dx.doi.org/10.1007/s10514-006-9043-2>.

GALINDO, C.; SAFFIOTTI, A.; CORADESCHI, S.; BUSCHKA, P.; FERNANDEZ-MADRIGAL, J. A.; GONZALEZ, J. Multi-hierarchical semantic maps for mobile robotics. In: **Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2005. p. 2278–2283. ISSN 2153-0858.

GALVEZ-LÓPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. **IEEE Transactions on Robotics**, v. 28, n. 5, p. 1188–1197, Oct 2012. ISSN 1552-3098.

GARAY-VITORIA, N.; ABASCAL, J. Text prediction systems: a survey. **Universal Access in the Information Society**, v. 4, n. 3, p. 188–203, 2006. ISSN 1615-5297. Available from Internet: <http://dx.doi.org/10.1007/s10209-005-0005-9>.

GRANSTROM, K.; CALLMER, J.; RAMOS, F.; NIETO, J. Learning to detect loop closure from range data. In: **Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2009. p. 15–22. ISBN 978-1-4244-2788-8. ISSN 1050-4729. Available from Internet: <http://dx.doi.org/10.1109/ROBOT.2009.5152495>.

GRISETTI, G.; KUMMERLE, R.; STACHNISS, C.; FRESE, U.; HERTZBERG, C. Hierarchical optimization on manifolds for online 2d and 3d mapping. In: **Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 273–278. ISSN 1050-4729.

GRISETTI, G.; KUMMERLE, R.; STACHNISS, C.; BURGARD, W. A tutorial on graph-based slam. **IEEE Intelligent Transportation Systems Magazine**, IEEE Press, Piscataway, NJ, USA, v. 2, n. 4, p. 31–43, winter 2010. ISSN 1939-1390.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. **IEEE Transactions on Robotics**, IEEE Press, Piscataway, NJ, USA, v. 23, n. 1, p. 34–46, feb. 2007. ISSN 1552-3098.

GRISETTI, G.; STACHNISS, C.; GRZONKA, S.; BURGARD, W. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In: **Proceedings of the 2007 Robotics: Science and Systems (RSS) Conference**. Cambridge, MA, USA: MIT Press, 2007.

GUENNEBAUD, G.; JACOB, B. et al. **Eigen v3**. 2010. Available from Internet: <http://eigen.tuxfamily.org>.

GUTMANN, J. S.; KONOLIGE, K. Incremental mapping of large cyclic environments. In: **Proceedings of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)**. Piscataway, NJ, USA: IEEE Press, 1999. p. 318–325.

HAHNEL, D.; BURGARD, W.; FOX, D.; THRUN, S. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In: **Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2003. v. 1, p. 206–211.

HAHNEL, D.; TRIEBEL, R.; BURGARD, W.; THRUN, S. Map building with mobile robots in dynamic environments. In: **Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2003. v. 2, p. 1557–1563. ISSN 1050-4729.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: **Proceedings of the 4th Alvey Vision Conference**. Sheffield, UK: University of Sheffield Printing Unit, 1988. p. 147–151.

HO, K. L.; NEWMAN, P. Detecting loop closure with scene sequences. **International Journal of Computer Vision**, v. 74, n. 3, p. 261–286, 2007. ISSN 1573-1405. Available from Internet: <http://dx.doi.org/10.1007/s11263-006-0020-1>.

HOWARD, A.; ROY, N. **The Robotics Data Set Repository (Radish)**. 2003. Available from Internet: <http://radish.sourceforge.net/>.

JOHANNSSON, H.; KAESS, M.; FALLON, M.; LEONARD, J. J. Temporally scalable visual slam using a reduced pose graph. In: **Proceedings of the 2013 IEEE International Conference on Robotics and Automation**. Piscataway, NJ, USA: IEEE Press, 2013. p. 54–61. ISSN 1050-4729.

JORGE, V. A. M. **Color Wideline Detector and Local Width Estimation**. Dissertation (Master) — Universidade Federal do Rio Grande do Sul, 2012.

JORGE, V. A. M.; MAFFEI, R.; FRANCO, G. S.; DALTROZO, J.; GIAMBASTIANI, M.; KOLBERG, M.; PRESTES, E. Ouroboros: Using potential field in unexplored regions to close loops. In: **Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2015. p. 2125–2131.

JULIER, S. J.; UHLMANN, J. K. A new extension of the kalman filter to nonlinear systems. In: **Proceedings of the 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls**. Bellingham, WA, USA: SPIE, 1997.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing (2nd Edition)**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2000. ISBN 0131873210.

KAESS, M.; JOHANNSSON, H.; ROBERTS, R.; ILA, V.; LEONARD, J. J.; DELLAERT, F. isam2: Incremental smoothing and mapping using the bayes tree. **The International Journal of Robotics Research**, 2011. Available from Internet: <http://ijr.sagepub.com/content/early/2011/12/19/0278364911430419.abstract>.

KAESS, M.; RANGANATHAN, A.; DELLAERT, F. isam: Fast incremental smoothing and mapping with efficient data association. In: **Proceedings of the 2007 IEEE International Conference on Robotics and Automation**. Piscataway, NJ, USA: IEEE Press, 2007. p. 1670–1677. ISSN 1050-4729.

KLEENE, S. C. Representation of events in nerve nets and finite automata. In: SHANNON, C.; MCCARTHY, J. (Ed.). **Automata Studies**. Princeton, NJ: DTIC Document, 1956. p. 3–41.

KONOLIGE, K.; BOWMAN, J. Towards lifelong visual maps. In: **Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2009. p. 1156–1163.

KOSTAVELIS, I.; GASTERATOS, A. Semantic mapping for mobile robotics tasks: A survey. **Robotics and Autonomous Systems**, v. 66, p. 86–103, 2015. ISSN 0921-8890. Available from Internet: <//www.sciencedirect.com/science/article/pii/S0921889014003030>.

KRAJNIK, T.; FENTANES, J. P.; CIELNIAK, G.; DONDRUP, C.; DUCKETT, T. Spectral analysis for long-term robotic mapping. In: **Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2014. p. 3706–3711. ISSN 1050-4729.

KUMAR, S.; GUIVANT, J.; DURRANT-WHYTE, H. Informative representations of unstructured environments. In: **Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2004. v. 1, p. 212–217. ISSN 1050-4729.

LACROIX, S.; MALLET, A.; BONNAFOUS, D.; BAUZIL, G.; FLEURY, S.; HERRB, M.; CHATILA, R. Autonomous rover navigation on unknown terrains: Functions and integration. **The International Journal of Robotics Research**, v. 21, n. 10-11, p. 917–942, 2002. Available from Internet: <http://ijr.sagepub.com/content/21/10-11/917.abstract>.

LEONARD, J. J.; DURRANT-WHYTE, H. F. Mobile robot localization by tracking geometric beacons. **IEEE Transactions on Robotics and Automation**, v. 7, n. 3, p. 376–382, Jun 1991. ISSN 1042-296X.

LIU, L.; ZHANG, D. Palm-line detection. In: **Proceedings of the 2005 IEEE International Conference on Image Processing (ICIP)**. Piscataway, NJ, USA: IEEE Press, 2005. v. 3, p. –269. ISSN 1522-4880.

LIU, L.; ZHANG, D.; YOU, J. Detecting wide lines using isotropic nonlinear filtering. **IEEE Transactions on Image Processing**, v. 16, n. 6, p. 1584–1595, June 2007. ISSN 1057-7149.

LIU, L. L.; ZHANG, D. Extracting tongue cracks using the wide line detector. In: ZHANG, D. (Ed.). **Medical Biometrics**. Berlin, Heidelberg: Springer, 2008. p. 49–56.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, Springer Netherlands, Dordrecht, Netherlands, v. 60, n. 2, p. 91–110, 2004. ISSN 0920-5691.

LU, F.; MILIOS, E. Globally consistent range scan alignment for environment mapping. **Autonomous Robots**, Kluwer Academic Publishers, v. 4, n. 4, p. 333–349, 1997. ISSN 0929-5593. Available from Internet: <http://dx.doi.org/10.1023/A%3A1008854305733>.

LU, F.; MILIOS, E. Robot pose estimation in unknown environments by matching 2d range scans. **Journal of Intelligent and Robotic Systems**, Springer, v. 18, n. 3, p. 249–275, 1997.

MACHARET, D. G.; NETO, A. A.; NETO, V. F. d. C.; CAMPOS, M. F. M. Efficient target visiting path planning for multiple vehicles with bounded curvature. In: **Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2013. p. 3830–3836. ISSN 2153-0858.

MAFFEI, R.; JORGE, V.; KOLBERG, M.; PRESTES, E. Segmented dp-slam. In: **Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2013. p. 31–36. ISSN 2153-0858.

MAFFEI, R.; JORGE, V. A. M.; PRESTES, E.; KOLBERG, M. Integrated exploration using time-based potential rails. In: **Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2014. p. 3694–3699.

MAFFEI, R.; JORGE, V. A. M.; REY, V. F.; KOLBERG, M.; PRESTES, E. Fast monte carlo localization using spatial density information. In: **Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2015. p. 6352–6358.

MAFFEI, R.; JORGE, V. A. M.; REY, V. F.; FRANCO, G. S.; GIAMBASTIANI, M.; BARBOSA, J.; KOLBERG, M.; PRESTES, E. Using n-grams of spatial densities to construct maps. In: **Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2015. p. 3850–3855.

MAFFEI, R.; JORGE, V. A. M.; REY, V. F.; KOLBERG, M.; PRESTES, E. Long-term place recognition using multi-level words of spatial densities. In: **Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2016. p. 3269–3274.

MAKARENKO, A. A.; WILLIAMS, S. B.; BOURGAULT, F.; DURRANT-WHYTE, H. F. An experiment in integrated exploration. In: **Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2002. v. 1, p. 534–539.

MANNING, C. D.; SCHÜTZE, H. **Foundations of Statistical Natural Language Processing**. Cambridge, MA, USA: MIT Press, 1999. ISBN 0-262-13360-1.

MASON, J.; MARTHI, B. An object-based semantic world model for long-term change detection and semantic querying. In: **Proceedings of the 2012 IEEE/RSJ International**

Conference on Intelligent Robots and Systems (IROS). Piscataway, NJ, USA: IEEE Press, 2012. p. 3851–3858. ISSN 2153-0858.

MILFORD, M. J.; WYETH, G. F. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In: **Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2012. p. 1643–1649. ISSN 1050-4729.

MILLAR, E.; SHEN, D.; LIU, J.; NICHOLAS, C. Performance and scalability of a large-scale n-gram based information retrieval system. **Journal of Digital Information**, v. 1, n. 5, 2006. ISSN 1368-7506. Available from Internet: <https://journals.tdl.org/jodi/index.php/jodi/article/view/22>.

MITTAL, A.; PARAGIOS, N. Motion-based background subtraction using adaptive kernel density estimation. In: **Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**. Piscataway, NJ, USA: IEEE Press, 2004. v. 2, p. –302. ISSN 1063-6919.

MONTEMERLO, M.; THRUN, S. **FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. (Springer Tracts in Advanced Robotics). ISBN 3540463992.

MONTEMERLO, M.; THRUN, S.; KOLLER, D.; WEGBREIT, B. Fastslam: A factored solution to the simultaneous localization and mapping problem. In: **Proceedings of the AAAI National Conference on Artificial Intelligence**. Edmonton, Canada: AAAI, 2002.

MORAVEC, H.; ELFES, A. High resolution maps from wide angle sonar. In: **Proceedings of the 1985 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 1985. v. 2, p. 116–121.

MOZOS, Ó. **Semantic Labeling of Places with Mobile Robots**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. (Springer Tracts in Advanced Robotics, v. 61). ISBN 978-3-642-11209-6.

MOZOS, O. M.; BURGARD, W. Supervised learning of topological maps using semantic information extracted from range data. In: **Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2006. p. 2772–2777. ISSN 2153-0858.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDÓS, J. D. Orb-slam: A versatile and accurate monocular slam system. **IEEE Transactions on Robotics**, v. 31, n. 5, p. 1147–1163, Oct 2015. ISSN 1552-3098.

MURPHY, R. R. **An Introduction to AI Robotics (Intelligent Robotics and Autonomous Agents)**. The MIT Press, 2000. ISBN 0262133830. Available from Internet: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262133830>.

NAUR, P.; BACKUS, J. W.; BAUER, F. L.; GREEN, J.; KATZ, C.; MCCARTHY, J.; PERLIS, A. J.; RUTISHAUSER, H.; SAMELSON, K.; VAUQUOIS, B. et al. Revised report on the algorithmic language algol 60. **Communications of the ACM**, v. 6, n. 1, p. 1–17, 1963.

NI, K.; DELLAERT, F. Multi-level submap based slam using nested dissection. In: **Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 2558–2565. ISSN 2153-0858.

NUCHTER, A.; LINGEMANN, K.; HERTZBERG, J.; SURMANN, H. 6d slam with approximate data association. In: **Proceedings of the 12th International Conference on Advanced Robotics (ICAR)**. Piscataway, NJ, USA: IEEE Press, 2005. p. 242–249.

NÜCHTER, A.; SURMANN, H.; LINGEMANN, K.; HERTZBERG, J. Semantic scene analysis of scanned 3d indoor environments. In: **Proceedings of the 8th International Fall Workshop on Vision, Modeling, and Visualization (VMV)**. Berlin, Germany: Akademische Verlagsgesellschaft Aka GmbH, 2003.

NÜCHTER, A.; WULF, O.; LINGEMANN, K.; HERTZBERG, J.; WAGNER, B.; SURMANN, H. 3d mapping with semantic knowledge: Robot soccer world cup ix. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 335–346. ISBN 978-3-540-35438-3. Available from Internet: <http://dx.doi.org/10.1007/11780519_30>.

OLIVEIRA, A. B. d.; SILVA, P. R. d.; BARONE, D. A. C. A novel 2d shape signature method based on complex network spectrum. **Pattern Recognition Letters**, v. 63, p. 43–49, 2015. ISSN 0167-8655. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167865515001749>.

OLSON, E. Recognizing places using spectrally clustered local matches. **Robotics and Autonomous Systems**, v. 57, n. 12, p. 1157–1172, 2009. ISSN 0921-8890. Inside Data Association. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0921889009001018>.

OLSON, E. M3rsm: Many-to-many multi-resolution scan matching. In: **Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2015.

OLSON, E.; LEONARD, J.; TELLER, S. Fast iterative alignment of pose graphs with poor initial estimates. In: **Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2006. p. 2262–2269. ISSN 1050-4729.

OLSON, E. B. Real-time correlative scan matching. In: **Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2009. p. 4387–4393. ISSN 1050-4729.

PAULS, A.; KLEIN, D. Faster and smaller n-gram language models. In: **Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (HLT '11), p. 258–267. ISBN 978-1-932432-87-9. Available from Internet: <http://dl.acm.org/citation.cfm?id=2002472.2002506>.

PETERSEN, K. B.; PEDERSEN, M. S. **The Matrix Cookbook**. Technical University of Denmark, 2012. Version 20121115. Available from Internet: <http://www2.imm.dtu.dk/pubdb/p.php?3274>.

PRESTES, E.; IDIART, M. Computing navigational routes in inhomogeneous environments using bvp path planner. In: **Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 1427–1432. ISSN 2153-0858.

PRESTES, E.; TREVISAN, M.; IDIART, M. A. P.; ENGEL, P. M. Bvp-exploration: further improvements. In: **Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2003. v. 4, p. 3239–3244.

PRONOBIS, A.; JENSFELT, P. Large-scale semantic mapping and reasoning with heterogeneous modalities. In: **Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2012. p. 3515–3522. ISSN 1050-4729.

REITHINGER, N.; ENGEL, R.; KIPP, M.; KLESEN, M. Predicting dialogue acts for a speech-to-speech translation system. In: **Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP)**. Piscataway, NJ, USA: IEEE Press, 1996. v. 2, p. 654–657.

ROSTEN, E.; PORTER, R.; DRUMMOND, T. Faster and better: A machine learning approach to corner detection. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 32, n. 1, p. 105–119, jan. 2010. ISSN 0162-8828.

ROTTMANN, A.; MOZOS, Ó. M.; STACHNISS, C.; BURGARD, W. Semantic place classification of indoor environments with mobile robots using boosting. In: **Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3**. AAAI Press, 2005. (AAAI'05), p. 1306–1311. ISBN 1-57735-236-x. Available from Internet: <http://dl.acm.org/citation.cfm?id=1619499.1619543>.

RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. Orb: An efficient alternative to sift or surf. In: **Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV)**. Piscataway, NJ, USA: IEEE Press, 2011. p. 2564–2571. ISSN 1550-5499.

RUSINKIEWICZ, S.; LEVOY, M. Efficient variants of the icp algorithm. In: **Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling (3DIM)**. Piscataway, NJ, USA: IEEE Press, 2001. p. 145–152.

RUSU, R. B.; MARTON, Z. C.; BLODOW, N.; DOLHA, M.; BEETZ, M. Towards 3d point cloud based object maps for household environments. **Robotics and Autonomous Systems**, v. 56, n. 11, p. 927–941, 2008. ISSN 0921-8890. Semantic Knowledge in Robotics. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0921889008001140>.

SACK, D.; BURGARD, W. A comparison of methods for line extraction from range data. In: **Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)**. Amsterdam, Netherlands: Elsevier Science, 2003.

SAEEDI, S.; PAULL, L.; TRENTINI, M.; SETO, M.; LI, H. Efficient map merging using a probabilistic generalized voronoi diagram. In: **Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2012. p. 4419–4424. ISSN 2153-0858.

SCOTT, D. W. **Multivariate Density Estimation: Theory, Practice, and Visualization**. Hoboken, NJ, USA: John Wiley & Sons, Inc., 1992. (Wiley Series in Probability and Statistics). ISBN 9780470316849.

SHANNON, C. A mathematical theory of communication. **Bell System Technical Journal**, ACM, v. 27, n. 1, p. 379–423, 1948.

SHANNON, C. E. Prediction and entropy of printed english. **Bell System Technical Journal**, v. 30, p. 50–64, jan 1951.

SICILIANO, B.; KHATIB, O. (Ed.). **Springer Handbook of Robotics**. Berlin, Heidelberg: Springer, 2008. ISBN 978-3-540-23957-4. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-30301-5>.

SIDOROV, G.; VELASQUEZ, F.; STAMATATOS, E.; GELBUKH, A.; CHANONA-HERNÁNDEZ, L. Syntactic n-grams as machine learning features for natural language processing. **Expert Systems with Applications**, v. 41, n. 3, p. 853–860, 2014. ISSN 0957-4174. Methods and Applications of Artificial and Computational Intelligence. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0957417413006271>.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X.

SILVERMAN, B. W. **Density estimation for statistics and data analysis**. 26. ed. London: Chapman & Hall, 1986. (Monographs on Statistics and Applied Probability, v. 26).

SMITH, R.; SELF, M.; CHEESEMAN, P. Estimating uncertain spatial relationships in robotics. In: COX, I. J.; WILFONG, G. T. (Ed.). **Autonomous Robot Vehicles**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1990. v. 8, chp. Autonomous robot vehicles, p. 167–193. ISBN 0-387-97240-4. Available from Internet: <http://dl.acm.org/citation.cfm?id=93002.93291>.

SMITH, S. M.; BRADY, J. M. Susan - a new approach to low level image processing. **International journal of computer vision**, Springer, v. 23, n. 1, p. 45–78, 1997.

SONG, J.; LI, Z. Sequential scan matching with sensor order. In: **Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2012. p. 4956–4961. ISSN 1050-4729.

STACHNISS, C.; GRISETTI, G.; BURGARD, W. Recovering particle diversity in a rao-blackwellized particle filter for slam after actively closing loops. In: **Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2005. p. 655–660.

STACHNISS, C.; HAHNEL, D.; BURGARD, W. Exploration with active loop-closing for fastslam. In: **Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2004. v. 2, p. 1505–1510.

STACHNISS, C.; MOZOS, O. M.; BURGARD, W. Speeding-up multi-robot exploration by considering semantic place information. In: **Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2006. p. 1692–1697. ISSN 1050-4729.

STAMATATOS, E. Plagiarism detection using stopword n-grams. **Journal of the American Society for Information Science and Technology**, Wiley Subscription Services, Inc., A Wiley Company, v. 62, n. 12, p. 2512–2527, 2011. ISSN 1532-2890.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 1st. ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN 1848829345, 9781848829343.

THRUN, S. A probabilistic on-line mapping algorithm for teams of mobile robots. **The International Journal of Robotics Research**, v. 20, n. 5, p. 335–363, 2001. Available from Internet: <http://ijr.sagepub.com/content/20/5/335.abstract>.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)**. Cambridge, MA, USA: MIT Press, 2005. (Intelligent robotics and autonomous agents). ISBN 9780262201629. Available from Internet: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262201623>.

THRUN, S.; LEONARD, J. J. Simultaneous localization and mapping. In: SICILIANO, B.; KHATIB, O. (Ed.). **Springer handbook of robotics**. Berlin, Heidelberg: Springer, 2008. p. 871–889.

THRUN, S.; MONTEMERLO, M. The graph slam algorithm with applications to large-scale mapping of urban structures. **The International Journal of Robotics Research**, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 25, n. 5-6, p. 403–429, 2006. Available from Internet: <http://ijr.sagepub.com/content/25/5-6/403.abstract>.

TIPALDI, G.; MEYER-DELIUS, D.; BEINHOFER, M.; BURGARD, W. Simultaneous localization and dynamic state estimation in reconfigurable environments. In: **IEEE/RSJ IROS Workshop on Metrics and Methodologies for Autonomous Robot Teams in Logistics (MMART-LoG)**. Piscataway, NJ, USA: IEEE Press, 2011.

TIPALDI, G. D.; ARRAS, K. O. Flirt - interest regions for 2d range data. In: **Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 3616–3622. ISSN 1050-4729.

TIPALDI, G. D.; MEYER-DELIUS, D.; BEINHOFER, M.; BURGARD, W. Lifelong localization and dynamic map estimation in changing environments. In: **Proceedings of the 2012 Robotics: Science and Systems (RSS) Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments**. Cambridge, MA, USA: MIT Press, 2012.

WALCOTT, A.; KAESS, M.; JOHANNSSON, H.; LEONARD, J. J. Dynamic pose graph slam: Long-term mapping in low dynamic environments. In: **Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2012.

WANG, C.-C.; THORPE, C. E.; THRUN, S.; HEBERT, M.; DURRANT-WHYTE, H. F. Simultaneous localization, mapping and moving object tracking. **I. J. Robotic Res.**, v. 26, n. 9, p. 889–916, 2007.

WANG, X.; MCCALLUM, A.; WEI, X. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In: **Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM 2007)**. Piscataway, NJ, USA: IEEE Press, 2007. p. 697–702. ISSN 1550-4786.

WEISS, G.; WETZLER, C.; PUTTKAMER, E. V. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In: **Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS). 'Advanced Robotic Systems and the Real World'**. Piscataway, NJ, USA: IEEE Press, 1994. v. 1, p. 595–601.

WERNER, F.; MAIRE, F.; SITTE, J.; CHOSET, H.; TULLY, S.; KANTOR, G. Topological slam using neighbourhood information of places. In: **Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2009. p. 4937–4942.

WERNER, F.; SITTE, J.; MAIRE, F. Topological map induction using neighbourhood information of places. **Autonomous Robots**, Springer US, v. 32, n. 4, p. 405–418, 2012. ISSN 0929-5593. Available from Internet: <http://dx.doi.org/10.1007/s10514-012-9276-1>.

WOLF, D. F.; SUKHATME, G. S. Mobile robot simultaneous localization and mapping in dynamic environments. **Auton. Robots**, Kluwer Academic Publishers, Hingham, MA, USA, v. 19, n. 1, p. 53–65, jul 2005. ISSN 0929-5593. Available from Internet: <http://dx.doi.org/10.1007/s10514-005-0606-4>.

WOLF, D. F.; SUKHATME, G. S. Semantic mapping using mobile robots. **IEEE Transactions on Robotics**, v. 24, n. 2, p. 245–258, April 2008. ISSN 1552-3098.

ZHANG, L.; ZAPATA, R.; LÉPINAY, P. Self-adaptive monte carlo localization for mobile robots using range finders. **Robotica**, v. 30, n. 02, p. 229–244, 3 2012. ISSN 1469-8668.