

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

MATHIAS FASSINI MANTELLI

**Exploiting organisational semantic
information in indoor environments for the
object search problem**

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

Advisor: Profa. Dra. Mariana Luderitz Kolberg
Coadvisor: Prof. Dr. Renan de Queiroz Maffei

Porto Alegre
June 2022

CIP — CATALOGING-IN-PUBLICATION

Mantelli, Mathias Fassini

Exploiting organisational semantic information in indoor environments for the object search problem / Mathias Fassini Mantelli. – Porto Alegre: PPGC da UFRGS, 2022.

143 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2022. Advisor: Mariana Luderitz Kolberg; Coadvisor: Renan de Queiroz Maffei.

1. Mobile robotics. 2. Object search. 3. Organisational semantic information. 4. Robotics perception. 5. Indoor environments. I. Kolberg, Mariana Luderitz. II. Maffei, Renan de Queiroz. III. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos André Bulhões

Vice-Reitora: Prof^ª. Patricia Pranke

Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves

Diretora do Instituto de Informática: Prof^ª. Carla Maria Dal Sasso Freitas

Coordenador do PPGC: Prof. Claudio Rosito Jung

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

*“You are seeking happiness.
Learn this lesson, once and forever,
that you will find happiness only by
helping others to find it!”*

— OUTWITTING THE DEVIL

ACKNOWLEDGEMENTS

My supervisors Prof. Mariana Kolberg and Prof. Renan Maffei have been endless sources of guidance during all these years. I have been under Prof. Mariana's supervision since I was a master's degree student, and I have grown and learned immensely since then. I am thankful for the two opportunities she gave me to learn not just about Mobile Robotics but many other exciting topics. Prof. Renan has kindly been my co-supervisor even before becoming a professor, when he was still a PhD candidate at Phi Group. I have always admired him for his skills, critical thinking and out of the box ideas. His passion for Phi Group and his dedication have motivated and inspired me to do the same during these years. It was with their supervision that this work came into existence. Thank you for the excellent cooperation and for all of the opportunities I was given to, besides boosting my technical skills, broaden my personal horizons during this incredibly instructive period. Besides, I would like to thank Prof. Edson Prestes, head of Phi Group, for the moments we shared in the lab and for how much he taught me throughout these years.

I would like to thank Prof. Jim Torresen for receiving me at his ROBIN lab in Norway. This opportunity to live abroad and exchange knowledge with other researchers was essential for me as a PhD candidate, and I am thankful for that. Many thanks go also to the ROBIN members for the great time I had in the group.

Many special thanks to two international friends, Robin Weiss and Sadegh Hosseinpour. Robin has not only helped me to grow personally with great conversations about essential aspects of life, but he also has encouraged me to study and live abroad until it came true. I should thank him for always believing in me. Our road trip through Europe was a memorable adventure, and it changed my life. Sadegh has been a special friend for a long time, and we had great moments in Norway that I will not forget. My life experience in Norway would be way more challenging without our volleyball training sections, pool games, and camping adventures.

I should also thank the members of Phi Group for the great time I had in our group. I enjoyed the atmosphere and their support. A special thank goes to a close friend Diego Pittol. He has been a partner in crime since my master's, and I have learned a lot with him. Our discussions and his attention to every detail contributed to completing this thesis. I would not be able to go through this unique adventure without him. I thank him for everything he has done for me, mainly for taking care of me during the challenging

ABSTRACT

Nowadays, the mobile robotics research community deals with different high-level tasks that require the robot to manipulate or interact with objects that may not be in the robot's field of view. To find an object in unknown environments, the robot needs to look for it while gaining information about the environment and making decisions in real-time, known as the object search (OS) problem. The research community has proposed different approaches for dealing with the OS problem, relying on the objects' color or 3D shape as visual cues to guide the search. However, this geometric information (i.e., color or size) limits the robot's perception and, consequently, the robot's performance during the search. Therefore, we propose two OS systems that exploit the advantages of semantic information inferred from the organisation of both the environment and objects. The first one relies on semantic information inferred from numbers in text signs found in the environment. The goal is to find a target door label. The use of organisational semantic information in this scenario allows the robot to reduce the search costs by avoiding not promising corridors to contain the target door label. The detected numbers are used to estimate either the search continues towards unknown parts of the environment, or carefully search in the already known parts. The second proposed OS system is based on the changes in the organisation and arrangement of objects over time in the environment. It observes the environment and gathers data from the objects' placement through the time by executing its recording mode. This recorded data is later used when the robot executes the requesting mode to search for the target object. Both systems were evaluated in different environments and compared against other OS approaches in simulated and real scenarios. Even though our systems do not depend on specific SLAM systems and object detection algorithms, we have used Gmapping and YOLO in our experiments, respectively. The results of the experiments support our systems' efficiency and demonstrate the improvement in the searching performance with the aid of organisational semantic information.

Keywords: Mobile robotics. Object search. Organisational semantic information. Robotics perception. Indoor environments.

Explorando Informações Semânticas em Ambientes Internos

RESUMO

Atualmente, a comunidade científica de robótica móvel está lidando com diferentes tarefas de alto-nível que requerem que o robô manipule ou interaja com objetos que podem não estar no campo de visão do robô. Para encontrar um objeto em um ambiente desconhecido, o robô precisa procurar por ele enquanto ganha informação sobre o ambiente e toma decisões em tempo-real, conhecido como o problema de busca por objetos (BPO). A comunidade de pesquisa propôs diferentes soluções para abordar o problema de BPO, se baseando na cor, tamanho ou no que existe ao redor dos objetos. Contudo, todas essas informações geométricas (como por exemplo cor ou tamanho) limita a percepção do robô e, por consequência, o seu desempenho durante a busca. Portanto, nós propomos dois sistemas de BPO que exploraram as vantagens de informações semânticas inferidas a partir da organização tanto do ambiente quanto dos objetos presentes. O primeiro se baseia em informações semânticas inferidas de números em placas de texto encontrados no ambiente. O objetivo é encontrar a placa de texto da porta alvo. O uso da informação semântica organizacional neste cenário permite que o robô reduza os custos da busca por evitar corredores não promissores para conter a placa de texto da porta alvo. Os números detectados são usados para estimar se busca continua em direção a regiões desconhecidas ou se realiza a busca cuidadosamente em regiões já conhecidas. O segundo sistema de BPO é baseado nas mudanças na organização e arranjo dos objetos ao longo do tempo no ambiente. Nosso sistema observa o ambiente e coleta dados do posicionamento dos objetos ao longo do tempo executando o seu modo de gravação. Os dados gravados são usados posteriormente quando o robô executa o modo de requisição para buscar pelo objeto. Ambos os sistemas foram avaliados em diferentes ambientes e comparados contra outros sistemas de BPO em simulação e ambiente real. Apesar dos nossos sistemas não dependerem de um sistema de SLAM ou algoritmo para detecção de objectos específicos, nós usamos o Gmapping e o YOLO nos nossos experimentos, respectivamente. Os resultados dos nossos experimentos confirmam a eficiência dos nossos sistemas e demonstram a melhora no desempenho da busca com o auxílio das informações semânticas organizacional.

Palavras-chave: Robótica móvel. Busca por objectos. Informação semântica. Percepção robótica. Ambientes internos.

LIST OF FIGURES

Figure 2.1	Fundamental problems in mobile robots and their state estimation.	25
Figure 2.2	Graphical model of the fundamental mobile robotics problems.	27
Figure 2.3	Frameworks of two popular text detection and recognition methodologies..	31
Figure 2.4	Text localization and recognition overview.	33
Figure 2.5	Recognition problems related to generic object detection.....	34
Figure 2.6	YOLO’s model detection as a regression problem.	35
Figure 2.7	Different kernel profiles shown in 2D and 2D/2D.	37
Figure 2.8	Computing kernel density estimates in four different positions of an image.	39
Figure 2.9	Example of the two fundamental requirements of exploration at the boundaries of a partial map.	40
Figure 2.10	Representation of part of a 2D grid map m	41
Figure 2.11	Exploration process with the potential field.	42
Figure 4.1	Example of our mapping and segmentation modules.....	59
Figure 4.2	Example of the Image Processing module processing two images	62
Figure 4.3	Demonstration of how the increasing angle $\theta_{inc}(S(m_i))$ is computed in a segment	66
Figure 4.4	Partial 2D map of the environment to show the importance of Growing Direction factor	67
Figure 4.5	Partial 2D map of the environment to explain the robot and door orien- tation factors.....	71
Figure 4.6	Partial 2D map of the environment to explain the distance factor.	72
Figure 4.7	Software setup used in the simulated experiments	74
Figure 4.8	The four maps used in the simulated experiments.....	76
Figure 4.9	Map used in the experiments with the real robot.....	85
Figure 4.10	The Pioneer 3DX robot used in the real environment experiment.	85
Figure 4.11	Step-by-step of the performance of our NSOS system in the physical environment.	87
Figure 5.1	One example of our proposed LSOS system operating in our custom made simulated environment.	91
Figure 5.2	Flowchart that explains how the recording mode from our LSOS system works.....	93
Figure 5.3	Flowchart that explains how the requesting mode from our LSOS sys- tem works.....	94
Figure 5.4	Example of the computed heat map given a set of detected objects.....	96
Figure 5.5	Example of the robot at the edge of the kernel, and yet not recognising the object.....	97
Figure 5.6	Example of the modified kernel with an acute angle, and the recognised object.....	98
Figure 5.7	Step by step of the space reduction performed by our OS system.	98
Figure 5.8	The difference between the normal kernel and its inverse.....	99
Figure 5.9	The seven-rooms environment created on Gazebo simulator and the Husky robot.....	102
Figure 5.10	The floorplan of the single-resident apartment from the HH106 dataset. .	102
Figure 5.11	The robot’s route for the Brute Force OS system.	103
Figure 5.12	Results of the three OS systems for both request search times consid- ering many different setups.....	107

Figure 5.13 Examples of the robot’s path for the search performed by the three OS systems in both locations A and H.....	108
Figure 5.14 Results of the three OS systems for the request search at midnight in a modified version of <i>Mobile-Inv</i>	109
Figure 5.15 Results of the three OS systems for both request search times considering many different setups.....	109
Figure 6.1 Jaci disinfecting a cirurgical room.....	117
Figure 6.2 Hospital environment used during in simulation during the development of Jaci’s autonomous system.....	119
Figure 6.3 Jaci’s sensor (a) in its front and left side, and (b) in its back and right side	121

LIST OF TABLES

Table 3.1	Table comparing OS and spatial-temporal works.	54
Table 4.1	Different scenarios used on our simulated tests.	75
Table 4.2	Results of the greedy and our NSOS systems in the <i>Normal</i> scenario.	78
Table 4.3	Results of the greedy and our NSOS systems in the <i>Inverse</i> scenario.	78
Table 4.4	Results of the greedy and our NSOS systems in the <i>Hotel</i> scenario.	79
Table 4.5	Results of the greedy and our NSOS systems in the <i>KTH</i> scenario.	80
Table 4.6	The average, standard deviations, and shortest path from the <i>Normal</i> scenario	80
Table 4.7	The average, standard deviations, and shortest path from the <i>Inverse</i> scenario	81
Table 4.8	The average, standard deviations, and shortest path from the <i>Hotel</i> scenario.	81
Table 4.9	The average, standard deviations, and shortest path from the <i>KTH</i> scenario.	81
Table 4.10	Comparison of the optimal solution (shortest path) of each goal-door from each scenario	82
Table 4.11	Human performance to the problem of OS system.	84
Table 5.1	Mug's position recorded in five setups by the recording mode of our pro- posal.	105
Table 5.2	Ground-truth of the mug's position for every setup according to the re- questing hours.	105
Table 5.3	The 59 days data from HH106 used by our OS system.	112
Table 5.4	The data from the last day of HH106 used to test our OS system.	113
Table 5.5	The human's presence estimation from our OS system.	113

LIST OF ABBREVIATIONS AND ACRONYMS

BVP	Boundary Value Problem
DS	Door Simulator
DSR	Domestic Service Robot
ER	Extremal Region
FoV	Field of View
FPS	Frames per Second
HIMM	Histogram In-Motion Mapping
IFR	International Federation of Robotics
KDE	Kernel Density Estimation
MCL	Monte Carlo Localization
OCR	Optical Character Recognition
OS	Object Search
PD	Population Division
PSR	Professional Service Robot
RBPF	Rao-Blackwellized Particle Filter
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue, Depth
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SLAM	Simultaneous Localization and Mapping
SR	Service Robot
UAV	Unmanned Aerial Vehicle
YOLO	You Only Look Once
LiDAR	Light Detection And Rangin

LIST OF SYMBOLS

\mathbf{x}_t	robot's pose at time step t . It is composed by a three dimensional vector containing x, y , which represents the position, and θ , which represents the orientation.
\mathbf{m}	map of the environment as a vector of N objects, $\mathbf{m} = (\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_N)^T$, describing the scenario. In 2D grid maps, with $1 \leq i \leq N$, a cell $\mathbf{m}_i = (x, y)^T$ is associated to a 2D position (x, y) . In this thesis, the index i refers to a random cell in \mathbf{m} , r to the cell where the robot is in \mathbf{m} , and k the cell at the center of a circular kernel.
\mathbf{m}_i^*	the candidate cell most promising to bring the robot to the goal-door.
\mathbf{u}_t	control data at instant t , and it corresponds to the change of state in the time interval $(t - 1; t]$.
\mathbf{z}_t	measurement made by the robot at instant t . The vector of all of them acquired at the same instant t is $\mathbf{z}_t = (\mathbf{z}_t^1, \mathbf{z}_t^2, \dots, \mathbf{z}_t^K)^T$.
$K(\cdot)$	circular kernel that computes the free space density of a group of cells. Its possible profiles are identified as uniform, $UK(\cdot)$, gaussian, $GK(\cdot)$, and inverted, $IK(\cdot)$.
d	Manhattan distance between two cells.
r	radius of $K(\cdot)$.
\mathbf{m}_j^{\ll}	smallest distance between two cells.
\mathbf{T}	subset of cells that are within the area of the kernel at any moment.
$Q(\cdot)$	function that tests whether a cell is free.
$\Psi(\cdot)$	function that computes the free space density.
$\Upsilon(\cdot)$	function that computes the map segmentation.
δ	a threshold that defines how many different sizes of free areas are considered by the segmentation function.
\mathbf{s}	segment from \mathbf{m} .
\mathbf{I}	image.
X	a set of data sample.

- L** list containing the recognized number from door signs.
- $S(\cdot)$ function that returns the nearest segment of a cell.
- $L(\cdot)$ function that returns the list of door signs from a segment.
- l door number.
- $\varphi(\cdot)$ probability distribution for a map cell, \mathbf{m}_i , where the target object is in \mathbf{m} .
- $\text{SF}(\cdot)$ semantic factor, the outcome of the combination of growing direction, $\varphi_g(\cdot)$, parity, $\varphi_p(\cdot)$, factors.
- $\text{GF}(\cdot)$ geometric factor, the outcome of the combination of robot orientation, $\varphi_r(\cdot)$, door orientation, $\varphi_o(\cdot)$, and closeness, $\varphi_c(\cdot)$, factors.
- C** vector of candidate cells, $\mathbf{C} = (\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_M)^T$, for finding the goal-door.
- $\theta(\cdot)$ function that returns the angle of the growing direction factor.
- SL, BL functions that counts the amount of door labels are smaller and larger than the goal-door, respectively.
- $\zeta(\cdot)$ function that measures the possibility of a segment to have door signs smaller or larger than the goal-door.
- $\gamma(\cdot)$ function that measures the difference angle between the increasing angle and the Voronoi angle.
- AL, DL functions that counts the amount of door labels have their parities alike or different than the goal-door, respectively.
- w_p threshold used to control the minimum amount of detected door signs.
- $H_r[\cdot]$ the door orientation histogram.
- λ_r a threshold used to define how many most recent robot's orientations are saved.
- $D(\cdot, \cdot)$ function that counts the number of cells between two other specific cells.
- h** 2D gri heat map of the environment as a vector of N objects, $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_N)^T$, describing the scenario.
- O** set of detected objects.
- o** detected object, composed by its position within **H**, its category, the hour it has been detected, and the robot's position during the detection, i.e., $\mathbf{o}_j = (o_j^p, o_j^c, o_j^h, o_j^r)$.

$W(\cdot, \cdot)$ function that measures the weight of the time difference.

U number of rows and columns of a grid that an image is divided into.

B number of bounding boxes predicted in each grid cell of an image.

C number of conditional class probabilities for each grid cell of an image.

CONTENTS

1 INTRODUCTION	17
1.1 Objective	19
1.2 Contributions of this Thesis	21
1.3 Outline	23
2 THEORETICAL BACKGROUND	24
2.1 The Basics of Mobile Robotics	24
2.2 Environment Perception	30
2.2.1 Text Localization and Recognition	31
2.2.2 Object Detection using 2D images	33
2.2.3 Free space segmentation in grid maps	36
2.3 Action in the environment	39
2.3.1 Exploration of environments.....	39
2.3.2 OS problem	42
3 RELATED WORK	44
3.1 Object Search approaches	44
3.2 Spatio-Temporal models	50
3.3 Semantic map and heat map	52
3.4 A discussion on the Related Works	52
4 NSOS: USING NUMBERS TO INTERPRET THE ORGANISATION OF UNKNOWN ENVIRONMENTS	55
4.1 Proposal and contributions	55
4.2 Basis for NSOS system	57
4.2.1 Mapping module	58
4.2.2 Map Segmentation module	59
4.2.3 Image Processing module	61
4.3 Semantic Planner of our NSOS system	62
4.3.1 Combining the geometric and semantic factors to estimate where to go	63
4.3.2 Growing Direction factor	65
4.3.3 Parity factor.....	68
4.3.4 Robot and Door Orientation factors.....	69
4.3.5 Closeness factor	70
4.4 Experiments and Results	73
4.4.1 Simulated Experiment Setup.....	74
4.4.2 Semantic and Greedy Object Search systems.....	75
4.4.3 Human participants performance in object searching task	82
4.4.4 Experiments Using a Physical Robot.....	85
4.5 Summary	86
5 LSOS: USING CHANGES IN THE ORGANISATION OF THE ENVIRONMENT OVER TIME	89
5.1 Proposal and contributions	89
5.2 LSOS system's search strategy	92
5.2.1 The Heat Map and the representation of the objects' presence	94
5.2.2 Inverted kernel $IK(\cdot)$	98
5.2.3 Goal computation in the request mode	99
5.3 Experiments and Results	100
5.3.1 Simulation and dataset	101
5.3.2 Two other object search systems to be compared against ours.....	103
5.3.2.1 Brute Force OS system	103

5.3.2.2 Last Seen OS system.....	103
5.3.3 Simulated Experiment Setup.....	104
5.3.4 Results and discussion from the simulated experiments.....	106
5.3.5 Person presence estimation with HH106 dataset.....	111
5.4 Summary.....	113
6 APPLICATION OF THE PROPOSED APPROACHES	115
6.1 Jaci against COVID-19 and hospital infections.....	115
6.2 Jaci and the environment organisation	117
6.2.1 Disinfecting a specific room	118
6.2.2 Disinfecting a specific object within a room.....	119
6.3 Summary.....	121
7 CONCLUSION AND FUTURE WORK	123
REFERENCES.....	127
APPENDIX A — RESUMO EXPANDIDO	137
A.1 Objetivo.....	139
A.2 Contribuições desta Tese	141
A.3 Organização	142

1 INTRODUCTION

Robots can be grouped into different classes depending on their function and the workplace they are designed for (KUMAR et al., 2005; ROBOTICS, 2012; HAIDEGGER et al., 2013). Among all the classes of robots, service robots (SRs), are robots that work semi or completely autonomously to perform useful services, excluding manufacturing operations (ROBOTICS, 2012). The SRs come in all different designs, as they may or may not be equipped with an arm structure, and even though most of them are mobile, they can also be fixed in place (GARCIA-HARO et al., 2020). The International Federation of Robotics (IFR) divides SRs into two subclasses based on their usability: *professional* and *personal/domestic* service robots (LITZENBERGER, 2018). Some examples of the professional service robots (PSRs) are defence robots (MARTINIC, 2014), farmer-assistants (VAKILIAN; MASSAH, 2017), medical (ABUBAKAR et al., 2020), and logistic (THAMRONGAPHICHARTKUL et al., 2020). Examples of domestic service robots (DSRs) include but are not limited to vacuum cleaners (FORLIZZI; DISALVO, 2006), lawn-mowers (BORINATO, 2017), food and beverage waiters (WAN et al., 2020), and elderly assistants (HERSH, 2015). The market for SRs has been regularly rising, and it is no surprise that there is an expectation that it will grow even further in the next few years (ALMEIDA; FONG, 2011; CHIANG; TRIMI, 2020). The decreasing cost of hardware components (processors, motor drivers, and sensors), the increasing energy density and lower cost of batteries, and the threats caused by outbreaks such as COVID-19 drive this expansion (CHIANG; TRIMI, 2020).

According to the Population Division (PD) of the United Nations, in 2015, there were 901 million people aged 60 or over, representing 12% of the global population (DIVISION, 2015). Besides, the PD projects that by 2030, the number of older adults in the world will reach 1.4 billion and 2.1 billion by 2050. Several policies to tackle the problems of population ageing have been proposed by several countries, including, for example, facilities for the elderly (LIN; CHEN, 2018; SEDDIGH et al., 2020). However, placing elderly people in facilities for retirement or even in nursing homes may cause some problems, such as physically, emotionally, and psychologically dependencies (THEURER et al., 2015). Additionally, some elderly do not voluntarily stay at nursing homes, preferring to spend their remaining years at their home where they have a more positive self-image than those who live in the nursing homes (KOK; BERDEN; SADIRAJ, 2015; LIN; CHEN, 2018). The increasing number of older people living at

home supports the need for DSRs to automate processes and tasks that may be tedious, inconvenient, or even challenging for older people (PAULIUS; SUN, 2019; TORRESEN; KURAZUME; PRESTES, 2020). In general, these sorts of robots can contribute to practical tasks for humans as robot assistants or robot companions, such as watching older adults concerning emergencies, reminding them to take their medicines, and searching, picking, and placing objects (SPRUTE et al., 2017; TORRESEN et al., 2018; PAULIUS; SUN, 2019).

Additionally, while some of the main motivations for deploying SRs have been elderly assistance and productivity improvement, the COVID-19 pandemic has brought a more critical purpose for them (CHIANG; TRIMI, 2020). PSRs can be deployed to perform a series of applications to provide contactless services, ensuring humans can practice social distancing (SEIDITA et al., 2021). Besides disinfecting indoor environments (MANTELLI et al., 2022), PSRs also have the potential to support the hospitality industry (ROSETE et al., 2020), and deliver medications and food (LEE et al., 2009; YANG et al., 2020). The use of SRs in logistic applications is relevant during such unusual scenarios. Some national organisations from the United States identified logistics as one of the broad areas where robotics can make a difference during outbreaks (SEIDITA et al., 2021).

In many example applications that we listed above, it is likely that SRs have to perform some searching tasks. Simple examples would be DSRs searching and picking objects for elderly with mobility restrictions and PSRs delivering packages to a specific spot in an unknown environment. Similar to humans in the context of object searching tasks, SRs should also not rely on the assumption that the object (or regions of interest) they are searching for is already within their field of view (FoV) (SJÖÖ; AYDEMIR; JENSFELT, 2012). Hence, they have to find the target object in large-scale environments based on primarily their visual sensors, known as object search (OS) problem (AYDEMIR et al., 2013). However, how does an SR find the target object that is not initially within its FoV? One way to address this problem is to make the SR perform a brute-force OS, in which it visits the whole environment following a predefined search route. Even though this strategy seems a straightforward solution, it does not efficiently solve the problem (RASOULI et al., 2020). The SR will eventually find it as long as the target object is within the environment. However, the searching process may be time-consuming due to the long distances travelled by the robot. Another more efficient solution is to consider a search strategy that incorporates information from both the environment and the target object, to

improve the searching performance. For example, such information could be the shape of the room for the environment (e.g. recognise a kitchen and then search for a plate) (AYDEMIR et al., 2011a), and the colour or class for the target object (RASOULI et al., 2020). The search strategy is one of the most critical parts of an OS approach, as it directly impacts the efficiency of an OS system (AYDEMIR et al., 2013). Therefore, it must be robust and effective regardless of the environment the SR is performing the search.

The research community has proposed valuable works related to the OS problem (EKVALL; KRAGIC; JENSFELT, 2007; SJÖÖ et al., 2009; SJÖÖ; AYDEMIR; JENSFELT, 2012; AYDEMIR et al., 2013; RASOULI et al., 2020). The problem is proven to be NP-Complete (TSOTSOS, 1992; YE; TSOTSOS, 2001), which means that the optimal search solution can be computed by approximation (SJÖÖ; AYDEMIR; JENSFELT, 2012), minimising the search cost as much as possible. In the case of SR performing OS tasks, such approximation could be computed with the aid of strong cues provided by the semantics of both the environment and other objects in the SR’s surroundings (SJÖÖ; AYDEMIR; JENSFELT, 2012). Semantics can be regarded as the high-level information inferred (or “perceived”) from the environment, including but not limited to names and categories of different objects, rooms and locations (VASUDEVAN et al., 2007; SJÖÖ; AYDEMIR; JENSFELT, 2012; LIU et al., 2016). Similarly, semantic maps encode not just the geometric and topological description of the environment but also its semantic interpretation, providing a friendly way for robots to communicate with humans (LIU et al., 2016). Then, when the SR’s system processes the robot’s sensor readings to infer further knowledge about the surroundings, it increases the level of abstraction of the environment over time (BARBER et al., 2018). The use of both semantic information and map in robotic applications enhances the robot’s autonomy and robustness, besides facilitating some challenging tasks (CESAR et al., 2016).

1.1 Objective

The objectives of this thesis are the followings

- *exploit the organisation of both the environment and objects to infer semantic search cues to address the OS problem:* we aim to demonstrate that organisational semantic information may help in the OS problem, providing a way to estimate which regions are more likely to contain the target object than others;

- *infer organisational semantic information from numbers recognized from door signs:* we argue that the arrangement of the numbers from door signs follow a certain pattern (and rules). Understanding how the numbers are arranged makes possible to estimate how promising a certain corridor is, and then, decide whether the search should continue in the current region or not. We proposed a Number-based Semantic OS system to test this argument;
- *infer organisational semantic information from changes of semi-dynamic objects:* we defend that objects are mainly moved by humans according to their daily routines and habits. It means that there are high chances that objects are moved in a particular pattern throughout a period of time, suggesting a repetition from time to time. Estimating such a pattern from human-object interaction may be useful for the OS problem, to estimate which places a target object may be in a given time of the day. We proposed a Long-term Semantic OS system to verify this idea;
- *perform a coarse-to-fine OS search in indoor environments:* we aim to propose a larger OS system that makes possible to a SR find a target room, and interact with the object within it. By combining the proposed OS systems we end up with a coarse-to-fine OS system, which first searches for the room by its number, and later looks for a target object.

We claim that, in general, our society is not randomly organised, and there are several patterns and rules we follow every day. It is no surprise that humans can improve their efficiency while performing daily tasks, like OS, if the environment is merely logically organised. Thus, they can save energy and time during such tasks. For example, most cities have their own rules for numbering the properties, although there is no universal rule for that. The inhabitants can study it to understand the local numbering pattern. Thus, they can estimate where a particular unknown building is in the city, even if they have never been there. On the contrary, when there are no written rules to specify the organisation of the environment, humans can understand them just by observing the environment for a while. Inspired by human behaviour in OS tasks, we are interested in making the SRs take advantage of the organisation of environments to improve their performance in the OS task.

We consider that the organisational semantic information could be inferred from the environment through the SR's sensor readings, and it could be used to help SRs in search tasks. Such semantic information is helpful to OS systems because it could be used as high-level search cues. Then, with a semantic OS system, the SR would not need

to search the whole environment to find the target object. The human reasoning process relies on several sorts of high-level search cues during searches, like labels and signs or the acknowledgment that other people may interact with the environment. In our daily life, we read signs, symbols, and labels to evaluate which direction we should go to find a specific room in an unknown environment. Another example would be someone who first checks whether the family's car is at home to then search for the car key. In particular, we focus on the organisational semantic information inferred from the organisation of both static and dynamic environments, like the labels and signs in the first example or the acknowledgement that other people may move objects.

1.2 Contributions of this Thesis

This thesis presents results (MANTELLI et al., 2021; MANTELLI et al., 2022) showing that semantic information inferred from the organisation of the environment can help SRs in the OS problem. Specially, we show that the use of organisational semantic information as search cues in the search strategy of OS systems can make the SR save resources by not visiting the whole environment. Besides, we show that the proper use of semantic information can improve the SRs' perception to perform high-level tasks, bringing them closer to humans. We also present a discussion on how our contributions could be adapted and deployed to Jaci (MANTELLI et al., 2022). It is a recently released autonomous sanitiser SR that aims to aid both the fight against COVID-19 and hospital infections due to bacterias and fungus contaminations.

The first contribution of this thesis is a semantic system that performs the OS concerning the room organisation (MANTELLI et al., 2021). It aims to find a specific room in an unknown environment based on the organisation of door labels. Although humans heavily rely on texts, characters, and symbols for accomplishing several tasks, the use of numbers from text labels as a data source is not very popular in robotics. Despite the low interest by the research community, we argue that numbers detected from text labels have a great potential for providing search clues and are often found in man-made environments. This idea came from human behaviour when searching for someone's office in an unknown building. More specifically, we are interested in investigating how the numbers from door labels in corridors could be used to estimate whether a corridor is promising for finding the target office. The search strategy relies on the arrangement of door labels in indoor scenarios, and then it estimates which corridor is more promising

for achieving the goal. It is important to highlight that even though our OS system is looking for a specific number, it is still looking for an object. That is because the number is recognized from a door sign, which is associated to a door, the object. Therefore, by saying that our OS system is searching for a number from a sign, what we really mean is that the goal is to find the door, the object, associated to the specific sign.

The scenario of our first contribution is considered static. The door signs arrangement do not change very often, as well as the position or amount of doors within a building. Hence, the parts of the environment that our system is relying on are static. If an SR has to perform object manipulation tasks in the room it has just found, it will probably have to search for the objects. However, in contrast to doors, objects inside a room are more likely to have their positions changed by someone, resulting in a more dynamic scenario. Hence, such a robot should perform an OS in a dynamic environment, and that is the context of our next contribution.

Our second contribution is a long-term semantic system that searches for a target object in dynamic unknown environments (MANTELLI et al., 2022). It assumes that some objects within the environment are not always static, i.e., that the organisation of the environment changes over time and is associated with people's activities. In this way, its goal is to incorporate a person's routine and habits into the search strategy and then make search estimations. This work aimed to model the semantic information of how objects are organised over time within an environment. Then, it uses this information to avoid making the SRs search for the target object in not promising regions. This idea came from observing how the objects are placed over time and that every person has their own singularities in terms of object placement.

Lastly, we finish the contributions of this thesis with a discussion about the possibility of deploying our OS systems in the SR Jaci (MANTELLI et al., 2022). This robot was build aiming help the fight against COVID-19 and hospital contamination in general. It is equipped with a set of UV lights and it autonomously disinfect indoor environments. However, it presents some limitations that could be overcome by our systems. We present our ideas on how this deployment could be done, along with the possible boost in efficiency that our systems may provide to Jaci.

1.3 Outline

The outline of this thesis is as follows. First, in Chapter 2, we introduce the theoretical background of this thesis, presenting some of the main problems in mobile robotics, along with the most popular approaches that deal with each problem. We also introduce the general concepts of the OS problem, which is the foundation of this thesis. Lastly, we review other techniques that are used throughout this document. In Chapter 3 we discuss the main works proposed by the research community regarding the OS problem, and give an overview about many other works that have been used as inspiration to the development of this thesis. In Chapters 4 and 5, we present both of our OS systems, Number-based Semantic OS and Long-term Semantic OS, in addition to the experimental results for both systems. In Chapter 6, we introduce a discussion about how our OS systems could be deployed in a real SR called Jaci (MANTELLI et al., 2022). Besides, we also discuss the benefits our systems provide to the SR in an environment disinfection application. Lastly, in Chapter 7, we draw the conclusions about the current work and discuss the future directions.

2 THEORETICAL BACKGROUND

In the previous chapter, we have argued that high-level tasks would benefit from exploiting the semantic information inferred from spatial and temporal organization of the environments. We have chosen the OS problem to explore this idea, which aims to estimate a target object's location in a large unknown environment, usually with a camera attached to a mobile SR. We believe investigating this problem can enlarge our understanding regarding the benefits of employing semantic information to expand the SR's perception.

This chapter presents a theoretical background detailing techniques used throughout this thesis. The OS problem requires the SR to map the unknown environment and to estimate its position simultaneously. Simultaneous Localization and Mapping (SLAM) systems fulfill these requirements, as it computes the state estimation and builds an environment representation. Hence, we address the basic concepts of such systems and mobile robotics in general, from the individual localization and mapping problems to how they are combined into the SLAM systems. Besides, we cover the generic and central formulation of OS problems, which is the basis for the works presented in Chapters 4 and 5.

2.1 The Basics of Mobile Robotics

Mobile SRs perform several tasks that require them to be aware of their positions in the environment and the position of obstacles to avoid collisions. In most realistic scenarios where robots are deployed, such information is not directly available. Hence, they have to estimate it with their sensors, which provide noisy data from the environment (THRUN; BURGARD; FOX, 2006).

The state estimation in mobile robotics can be summarized in four variables:

- \mathbf{x}_t : robot's pose at time step t . It is composed by a three dimensional vector containing $(x, y, \theta)^T$, in which x, y represent the position and θ the orientation. A sequence of robot's poses from time step 0 to time step t is defined as $\mathbf{x}_{0:t} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_t\}$.
- \mathbf{m}_i : object i 's position in the environment. A list of N objects, with $1 \leq i \leq N$, in the environment along with their properties is given by the vector $\mathbf{m} = (\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_N)^T$.
- \mathbf{u}_t : control data at instant t . It corresponds to the change of state in the time interval

$(t - 1; t]$. The sequence of control data that takes the robot from the initial position to x_t is denoted by $\mathbf{u}_{1:t} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_t\}$.

- z_t^i : the i -th measurement made by the robot at instant t . The vector of all of them acquired at the same instant t is $z_t = (z_t^1, z_t^2, \dots, z_t^K)^T$, whereas $z_{1:t} = \{z_1, z_2, \dots, z_t\}$ expresses the history of all observations.

After defining the four variables that are the basic foundation for state estimation in mobile robotics, it is worth to explain their role in different estimation problems. The set of controls $\mathbf{u}_{1:t}$ and measurements $z_{1:t}$ are always known since the robot's sensors provide them. Inertial measurement units and wheel encoders are examples of sensors that provide control data, whereas LiDARs, sonars, and cameras measure the environment. The other two variables, robot's pose, $x_{0:t}$, and environmental map, m , are not necessarily known. Depending on the estimation problem, it is necessary to estimate different variables, like the three examples depicted in Figure 2.1. In *Localization*, Figure 2.1a, the map is known in advance, and hence, only the SR's pose is estimated. The opposite happens in *Mapping*, Figure 2.1b, as the map is built based on the known SR's pose. Lastly, in *SLAM*, Figure 2.1c, which combines the two previous problems, none of them is given a priori, and therefore, both are estimated simultaneously.

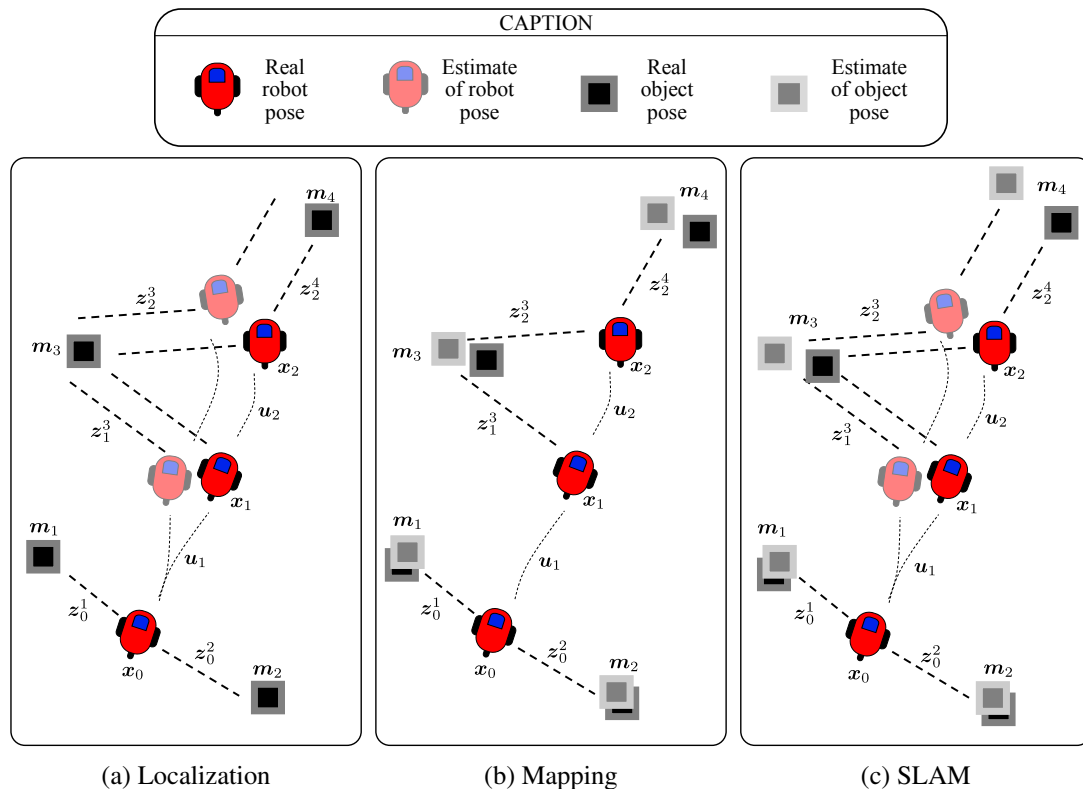


Figure 2.1 – Fundamental problems in mobile robots and their state estimation. It is estimated: (a) robot's pose, (b) map, (c) both of them simultaneously. Extracted from (MAFFEI, 2017).

Localization is the most basic perceptual problem in robotics. It aims to determine the SR's pose relative to a given map of the environment. Localization can also be seen as a problem of coordinate transformation, in which it is established a correspondence between the map coordinate system and the SR's local coordinate system (THRUN; BURGARD; FOX, 2006). There are multiple localization problems, and they are not equal in terms of their difficulty level. One characteristic that divides this problem into local and global localization is the awareness of the SR's initial pose. The former assumes that the initial SR's pose is known. Therefore, the problem becomes a sort of position tracking in which the noise in the measurements is adjusted in robot motion, commonly by a Gaussian distribution. On the other hand, the latter is unaware of the initial pose, making it perform the localization globally (where the name comes from) in the map. The global localization has a higher difficulty level than the local one, and one of its variations is even more challenging, the kidnapped robot problem. It addresses the problem of a localized robot being teleported to some other location in that the robot might believe it knows where it is while it does not. Although a SR is rarely kidnapped in practice, recovering from localization failures is essential for autonomous robots.

The formulation of the global localization problem is presented in Figure 2.2, which depicts a few iterations of the robot's pose estimation and how the variables are used. The map \mathbf{m} is already known, whereas the $\mathbf{x}_{0:t}$ must be estimated based on the controls $\mathbf{u}_{1:t}$ and the measurements $\mathbf{z}_{1:t}$. For the case of local localization, the \mathbf{x}_0 is known and hence, it does not need to be estimated. Markov localization is a probabilistic algorithm that addresses all the localization problems mentioned earlier. It applies the Bayes filter, $p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{m})$, to transform a probabilistic belief at time $t - 1$ into a belief at time t .

Many other localization algorithms implement Markov localization in mobile robotics. Three of them have been in the spotlight for a long time and are prevalent in this field: Kalman filter, grid-based filter, and particle filter. The former filters predicts in linear dynamics and measurement functions (LEONARD; DURRANT-WHYTE, 1991), whereas the grid-based filter approximates the estimations by decomposing the state space into finitely many regions of the grid map (BURGARD et al., 1998). The key idea of the latter, particle filter, is to represent the estimation by a set of random state samples, called particles, drawn from the previous estimation. It can represent a much broader space of distribution, in contrast to the Kalman filter that is more strict to Gaussians (DELLAERT et al., 1999). The particle filter implementation for mobile robotics is also known as

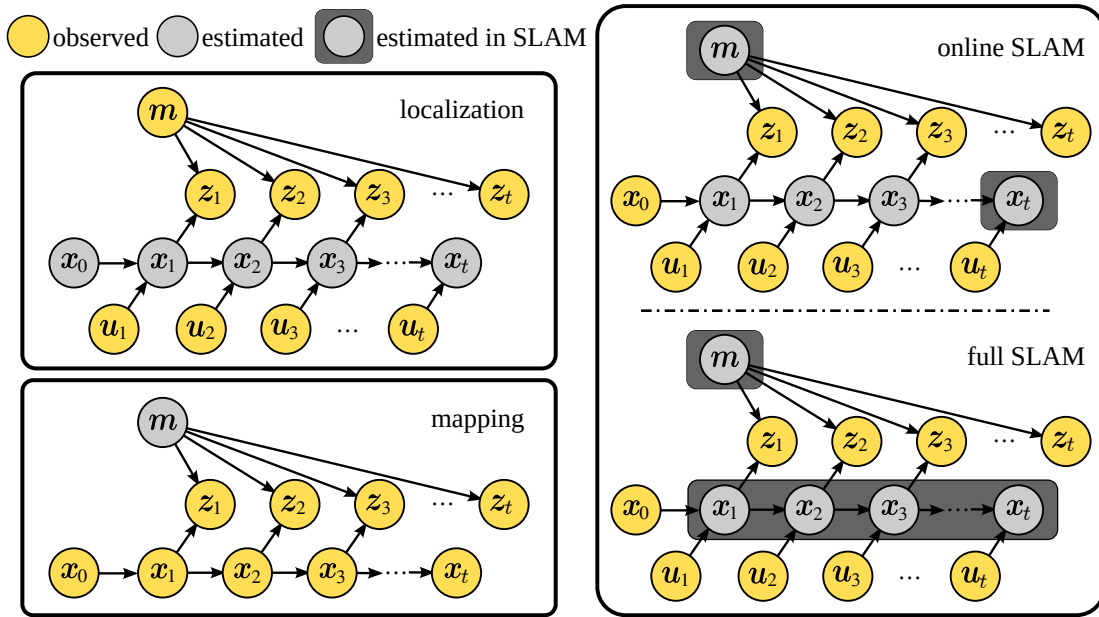


Figure 2.2 – Graphical model of the fundamental mobile robotics problems: localization, mapping, and two SLAM variations. Adapted from (MAFFEI, 2017).

Monte Carlo Localization (MCL), widely used in many different robotics applications for multiple robot types (DELLAERT et al., 1999; THRUN; BURGARD; FOX, 2006).

Mapping, for the case of the robot’s poses are known, is the problem of generating consistent maps from noisy and imprecise measurement data, as shown in Figure 2.1b (THRUN; BURGARD; FOX, 2006). The estimated belief of the map, $p(\mathbf{m} \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t})$, considers the set of all measurements up to time t , $\mathbf{z}_{1:t}$, along with the robot’s path defined by its history of all poses, $\mathbf{x}_{1:t}$, as also shown in Figure 2.2. Comparing the graphical models of the localization and mapping problems in Figure 2.2, one can say that they are opposite in terms of which variable each estimates. This thought makes sense, since whereas the former relies on \mathbf{m} to estimate $\mathbf{x}_{0:t}$, the latter relies on $\mathbf{x}_{0:t}$ to estimate \mathbf{m} . It is important to mention that the controls $\mathbf{u}_{1:t}$ play no role in this context, as the path is already known. Besides, the robot’s initial pose \mathbf{x}_0 is omitted from the map estimation because no measures are taken when the robot is at that pose.

Similar to the localization problem that groups multiple localization types, the mapping problem also represents a general idea implemented by different map types. The feature-based maps represent the cartesian location of features, which are distinct objects in the physical world, extracted from the measurements, such as images from visual sensors or a vector of distances from a 2D LiDAR (SALAS-MORENO et al., 2013; ENGEL; SCHÖPS; CREMERS, 2014; MUR-ARTAL; MONTIEL; TARDOS, 2015). The advantage of such a map type is the reduction of computational complexity, as the fea-

ture space has a lower dimension than the raw measurement. For example, the eight 3D edges of a boundingbox encircling a car are computationally cheaper to process than a point cloud from a 3D LiDAR. Another map type within the mapping problem is called location-based. It represents in each map component \mathbf{m}_i the regions from the environment, regardless of whether they contain objects. This way, any location in the world has a label on the map, not only features. Occupancy grid maps are often considered the most popular location-based map (THRUN; BURGARD; FOX, 2006). They discretize the environment into small portions called grid cells, which store information about the area it covers. In general, this information in each cell is a single value representing the probability that an obstacle occupies this cell. The size of the cells defines the map resolution, which brings a tradeoff between the level of details and the demand for memory resources.

Lastly, SLAM, also known as Concurrent Mapping and Localization, is undoubtedly the most fundamental and challenging problem in robotics (THRUN; BURGARD; FOX, 2006). SLAM problems appear in scenarios where the environmental map is unavailable and the robot is unaware of its pose, as depicted in Figure 2.1c. In contrast to the other two problems presented earlier, which have to estimate either the map \mathbf{m} or $\mathbf{x}_{1:t}$, in SLAM problems, the robot has to perform the estimation of both variables at the same time, as shown in Figure 2.2. Since the robot does not know its pose and there is no map, the pose \mathbf{x}_0 is assumed, by convention, to be $(0, 0, 0)^T$. The high difficulty level of SLAM comes from the double dependency of localization and mapping: to estimate the pose, the robot needs a map from the environment, whereas to estimate the map, the robot need to know its pose.

The SLAM problem is divided into two forms based on what is estimated: online and full SLAMs. The former focus on estimating only the posterior over the current robot's pose \mathbf{x}_t and the map \mathbf{m} , $p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$. The full SLAM computes the same estimation, but with the entire robot's trajectory $\mathbf{x}_{1:t}$ along with the map \mathbf{m} , $p(\mathbf{x}_{1:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$.

The majority of the algorithms for the online SLAM problem are incremental, i.e., the idea is to estimate the posterior probability on the current robot state and map as the robot moves, discarding past measurements and controls once they have been processed. The Kalman and particle filters are also used in this context, besides the localization one as previously discussed. The Extended Kalman Filter is the basis of one of the earliest online SLAM approaches, linearizing motion and observation models, which usually

are nonlinear, to perform the online SLAM estimations (MAFFEI, 2017). An online SLAM problem that is based on particle filter is known as Rao-Blackwellized particle filter (RBPF) (MURPHY et al., 1999; DOUCET et al., 2000; GRISETTIYZ; STACHNISS; BURGARD, 2005; GRISETTI; STACHNISS; BURGARD, 2007a). In RBPF, each particle carries an individual grid map of the environment, representing a hypothesis of the robot’s trajectory. The number of particles is directly related to the map quality since the higher this number, the broader is the hypotheses variety. However, there is a cost associated with each particle, and hence, it is not practical to increase the number of particles until the estimated map matches the physical world.

The algorithms for the full SLAM problem calculate a posterior over the entire path, which solves an issue in the online SLAM problem. Discarding the previous states after estimating the current one, also known as Markov assumption, implies that the possible poor estimations in the past are not adjustable. In contrast, the full SLAM problems backpropagate to the previous estimations the error reduction computed in the current state calculation. GraphSLAM captures the essence of the full SLAM problem, since it calculates a solution for the offline problem over $x_{1:t}$ and $z_{1:t}$ in m . Despite the advantage of improving previous state estimations, full SLAM algorithms are computationally heavy due to the optimization of nonlinear quadratic constraints.

Explaining the fundamental problems of mobile robotics, from the simplest localization to the more complex SLAM problems, helps to understand why the OS works for unknown environments depend on a SLAM system. Since our works in the next chapters are designed for similar conditions (large and unknown environments), we opted to rely on GMapping (GRISETTI; STACHNISS; BURGARD, 2007a). It is an online SLAM algorithm based on RBPF that provides a 2D grid map, and each cell contains a value that means whether the region it represents is unknown (to the SLAM system), occupied (obstacle), or free.

The aforementioned problems belong to the perception group in mobile robotics. The localization, mapping, and SLAM, use the robot’s perception to make estimations about the robot’s and world’s state, regardless their actions during the computation of the estimate. However, in mobile robotics there is a second large group of problems, the action. In contrast to the first group, perception, the problems within this group are responsible for choosing the right actions for the robot according to a policy (or a cost function) (THRUN; BURGARD; FOX, 2006). For example, an unmanned aerial vehicle (UAV) has to flight from point A to point B, and there are two possible paths. The first

one is shorter, but the UAV has to flight through many mountains, risking to colide and getting lost by losing its signal connection with satellites. The second path is longer, cir-cunventing the mountains in a safe way. Then, should the UAV take the risk and maybe land on point B as fast as possible, or delay its arrival and ensure it will arrive? Action selection in many robotics tasks is tied to the notion of uncertainty. In some cases, reducing uncertainty is the direct goal of action selection, like when a robot is exploring an environment. In other cases, reducing uncertainty is a way to achieving other goal, such as reliably arrinving at a target location (THRUN; BURGARD; FOX, 2006).

We presented the localization and the SLAM problems in this section, and they also have their own variations for action selection. In the active localization problem, an algorithm controls the robot to minimize the localization error arising from moving a poorly localized robot into a hazardous place (THRUN; BURGARD; FOX, 2006). The algorithm could avoid homogeneous regions in the environment, e.g. long corridors or squared empty rooms, where the localization error usually increases, and prefer other type of regions with distinguishable landmarks. For the SLAM problems, the action selection is performed by exploration techniques. Their combination resulted in another problem called integrated exploration, or active SLAM. The idea is to find a balance between exploring the unknown parts of the environment to gain information and returning to the visited parts to reduce the uncertainty regarding both the map and the robot's pose. The exploration alone is within the action group, and as our thesis is related to this topic, we will present more details about it further in this chapter.

2.2 Environment Perception

Different applications performed by robots depend on distinct information about the environment. The robot's sensors readings, such as images and point clouds, are input to various perception algorithms, like object detection or a QR code reader. For our thesis, we also depend on some environmental information that are perceived by the techniques we present below.

2.2.1 Text Localization and Recognition

Text can be embedded into documents or scenes as a mean of communicating information, and it is considered one of the most expressive means of communications (YE; DOERMANN, 2014). The process of Optical Character Recognition (OCR) aims to detect and recognize text in printed materials, images or videos, to then convert it into a digitized form so that machines can manipulate the digital text (YE; DOERMANN, 2014; ISLAM; ISLAM; NOOR, 2017). The OCR process has been in the spotlight for several years (ISLAM; ISLAM; NOOR, 2017). The motivation for such attention from the research community is that text provides meaningful information to be used in many applications. Besides, OCR is a complex problem, and constantly a new work is proposed to deal with the variety of languages, fonts, and styles in which text can be written, along with the complex rules of languages (ISLAM; ISLAM; NOOR, 2017).

There are two methodologies commonly used in OCR systems, *integrated* and *stepwise* (YE; DOERMANN, 2014). The integrated methodology recognizes words when the detection procedures share information with character classification and relies on joint optimization strategies, as shown in Figure 2.3a. On the other hand, stepwise methodology has separated detection and recognition modules. Besides, it uses a feed-forward pipeline to detect, segment, and recognize text regions, shown in Figure 2.3b (YE; DOERMANN, 2014). Lastly, it relies on a feedback procedure from text recognition to text detection to reduce false detections.

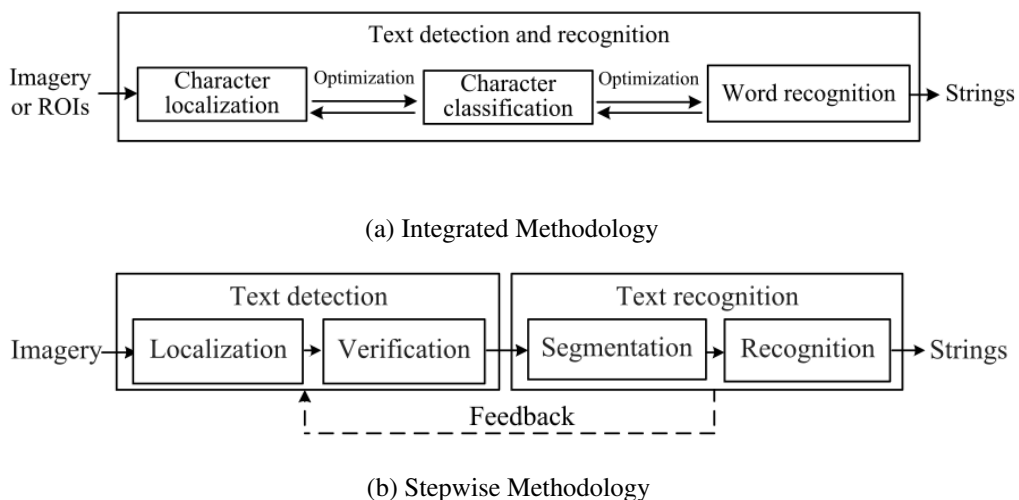


Figure 2.3 – Frameworks of two popular text detection and recognition methodologies, where (a) is the Stepwise methodology, and (b) is the Integrated methodology. Adapted from (YE; DOERMANN, 2014).

The latter methodology usually employs a coarse-to-fine strategy, which is done in

four steps: localization, verification, segmentation, and recognition (YE; DOERMANN, 2014). The goal of the first step, localization, is to coarsely classify components and groups into candidate text regions. In the second step, verification, such regions are further classified into text or non-text regions. The third step, segmentation, separates the characters in a way that exclusive, accurate outlines of image blocks remain for the recognition step. Lastly, the recognition step converts image blocks into characters. It is also possible that some stepwise methodologies ignore the verification and segmentation steps. Further, another adaptation is the inclusion of additional steps to perform text enhancement and rectification (YE; DOERMANN, 2014).

One of the leading methods in scene text detection is based on detecting characters, and the work proposed by Neumann and Matas (2012) is an example (ZHANG et al., 2016). It is an end-to-end real-time scene text localization and recognition method based on the stepwise methodology (NEUMANN; MATAS, 2012; YE; DOERMANN, 2014). Neumann and Matas addressed the character detection problem as an efficient sequential selection from the set of Extremal Regions (ERs) to achieve real-time performance. Such ER is a character detector that analyses image regions whose outer boundary pixels have higher values than the region itself. It is robust to blur, illumination, colour, and texture variation. Additionally, it also handles low-contrast text (NEUMANN; MATAS, 2012). The pixels within each ER's bounding box are described by a class of region descriptors that serve as features for the character classification.

In a given grayscale image, Figure 2.4b, the authors do the text localization by estimating the probability of each ER being a character using a set of features, Figure 2.4c. The verification step is done by selecting only the ERs from the previous step with locally maximal probability (of being a character), Figure 2.4d. Since the verification step aims to eliminate not promising character candidates, the authors further classify the high-likely ERs with more computationally expensive features to improve the character classification. Then, they group the verified ERs into words and select the most probable character segmentation, Figure 2.4e. The grouping is computed with a highly efficient exhaustive search with feedback loops. To get the text detected, Figure 2.4f, the average run time of the proposed method on a 800×600 image is 0.3s on a standard PC (NEUMANN; MATAS, 2012). The method proposed by Neumann and Matas achieved state-of-the-art text localization results when evaluated in two public datasets. Besides, they were the first ones to report results for the end-to-end text recognition on the ICDAR 2011 Robust Reading competition dataset (NEUMANN; MATAS, 2012). Due to the efficiency and ro-

business of the work proposed by proposed by Neumann and Matas (2012), we are using it in our work presented Chapter 4.

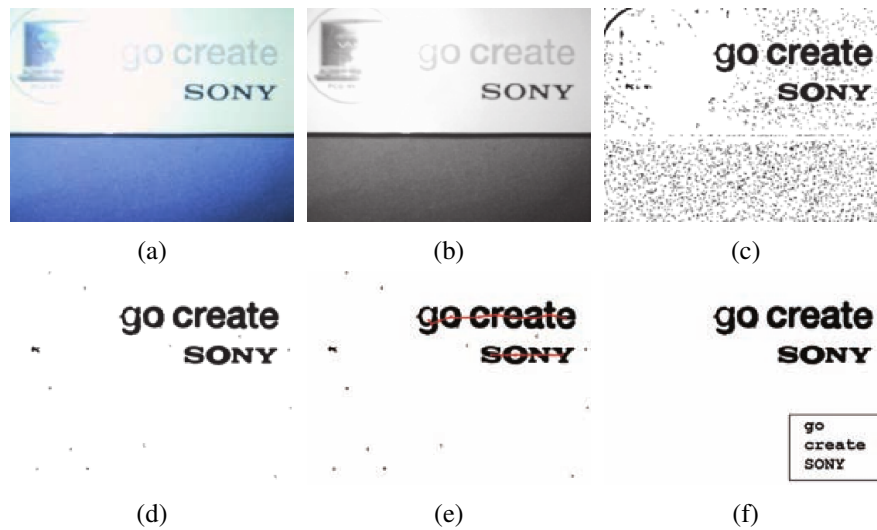


Figure 2.4 – Text localization and recognition overview, in which (a) is the source image, (b) is the extracted intensity channel, (c) is the ERs selected by the first stage of the sequential classifier, (d) is the ERs selected by the second stage of the classifier, (e) is the text lines found by region grouping, and (f) is the only ERs in text lines selected and text recognized by an OCR module. Extracted from (NEUMANN; MATAS, 2012).

2.2.2 Object Detection using 2D images

Generic object detection problem is defined as determining whether there are instances of objects from predefined categories in a given image. For the present instances, it is also necessary to return their spatial location and extent (LIU et al., 2020). This problem is also known as generic object category detection, object class detection, or even object category detection (LIU et al., 2020). It focuses on detecting a broad range of natural classes instead of specific object class detection, where only a narrower predefined class of interest may be present, such as faces, pedestrians, or cars.

Nowadays, the research community is more interested in detecting highly structured objects, like cars, bicycles, and airplanes, and articulated objects, such as humans and pets, rather than unstructured scenes (e.g., sky, grass, and cloud) (LIU et al., 2020). Regarding the spatial location and extend of an object within an image, like the detected objects in Figure 2.5a, it can be defined coarsely using a bounding box (EVERINGHAM et al., 2010; REDMON et al., 2016), which is a rectangle tightly bounding the object, Figure 2.5b, a precise pixel-wise segmentation masks (ZHANG et al., 2013), Figure 2.5c, or a closed boundary (RUSSELL et al., 2008; LIN et al., 2014), Figure 2.5d (LIU et al.,

2020).

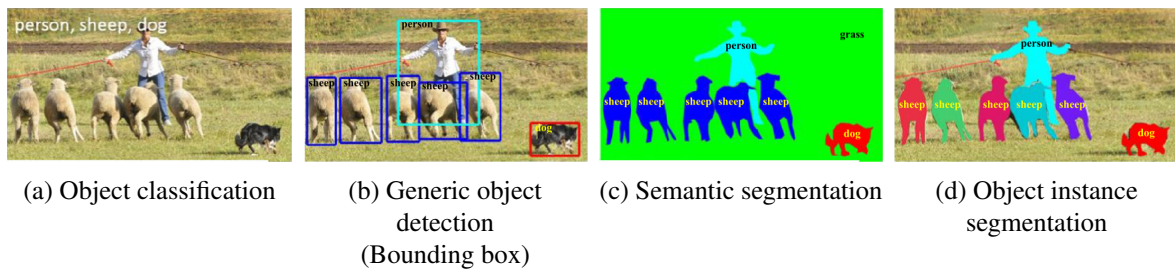


Figure 2.5 – Recognition problems related to generic object detection. This figure shows (a) image level object classification, (b) bounding box level generic object detection, (c) pixel-wise semantic segmentation, and (d) instance level semantic segmentation. Extracted from (LIU et al., 2020).

Recently, deep learning-based techniques have been proposed to deal with the object detection problem (REDMON et al., 2016). Deep learning has been used to solve many other challenging tasks, in areas such as image classification (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; HE et al., 2016) and language modeling (GONG et al., 2018; DAI et al., 2019)(HE; ZHAO; CHU, 2021). Deep learning techniques have arisen as powerful techniques for learning feature representations automatically from data (LIU et al., 2020). The success of deep learning in general in these areas is partly due to the rapidly developing of computational resources, such as powerful graphical cards, the availability of big training data, and the effectiveness of deep learning to extract hidden representations from images, texts, and videos (WU et al., 2020b).

A popular deep learning approach for object detection that has been improved since its first version is called YOLO (REDMON et al., 2016). The name is an acronym and is short for “*You Only Look Once*”, which partially explains the general idea of the approach. The YOLO’s authors have entirely abandoned the initial object detection paradigm of “*proposal detection + verification*”. Instead, they followed an entirely different idea: to apply a single neural network to the whole image. This idea made YOLO the first one-stage object detector in the deep learning era (where the name comes from) (ZOU et al., 2019). Thanks to this unified approach, YOLO detects objects extremely fast, processing images in real-time at 45 frames per second (FPS) (REDMON et al., 2016). To compare, the best object detector before YOLO, called Faster RCNN, processes images at 5 to 7 FPS (LIU et al., 2020).

Redmon et al. (2016) approached object detection as a regression problem, straight from image pixels to bounding box coordinates and class probabilities (LIU et al., 2020). Each image is evaluated once in their system by a single neural network that predicts bounding boxes and class probabilities. Due to this single network detection pipeline,

YOLO can be optimized end-to-end directly on detection performance (REDMON et al., 2016).

YOLO divides an image into an $U \times U$ grid, and each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object. In addition, the score also suggests how accurate YOLO thinks the box is what it predicts. In the second image of Figure 2.6, the score of the bounding boxes is represented by their thickness. The thicker the bounding box, the higher the confidence that there is an object in that location. Here is important to mention that at this point, YOLO does not know which objects there are in the image. It just knows whether exist any and their locations. To find out the object classes within the image, YOLO predicts C conditional class probabilities for each grid cell. These probabilities are conditioned on the grid cell containing an object. It means that YOLO is predicting that if there is an object in a cell, that object is an instance of the class with the highest probability. YOLO only predicts a set of class probabilities per grid cell regardless of the number of bounding boxes B for each grid cell. The third image of Figure 2.6 illustrates this prediction. Finally, YOLO multiplies the conditional class probabilities and the individual bounding box confidence predictions at the test time, which results in the class-specific confidence scores for each box. Hence, these scores encode both the probability of that class appearing in the bounding box and how well the predicted box fits the object, represented by the fourth image of Figure 2.6. Through a non-max suppression process, YOLO discards low-value bounding boxes and duplicated detections, resulting then in the final detections, illustrated by the fifth image of Figure 2.6.

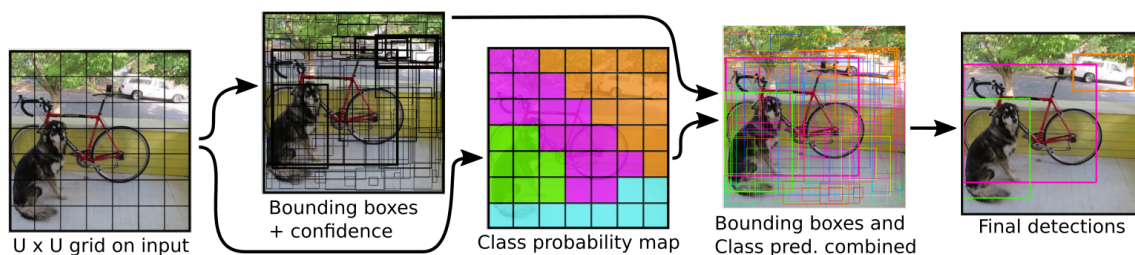


Figure 2.6 – YOLO’s model detection as a regression problem. The input image is initially divided into an $U \times U$ grid, and for each grid cell YOLO predicts B bounding boxes, confidence for those boxes, and C class probabilities. Adapted from (REDMON et al., 2016).

Unlike sliding window and region proposal-based techniques, YOLO reasons globally about the image when making predictions, processing the entire image during training and test time (REDMON et al., 2016). Hence, it implicitly encodes contextual information about classes and their appearance. Besides, YOLO learns generalizable representations of objects. In one of its experiments, YOLO was trained on natural images and

tested on artwork, outperforming top detection methods proposed by the research community (REDMON et al., 2016). However, despite its significant improvement in detection speed, YOLO suffers from a slight drop in localization accuracy compared with two-stage detectors, especially for some small objects (ZOU et al., 2019). The imposing performance of YOLO, proposed by Redmon et al. (2016), in detecting objects on 2D images convinced us to use it in our work presented Chapter 5.

2.2.3 Free space segmentation in grid maps

The process of computing the probability density function f of any data sample $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_n\}$ is called density estimation (DEVROYE; KRZYŻAK, 1999; MAFFEI, 2017; BHATTACHARJEE; GARG; MITRA, 2021). The kernel density estimation (KDE), $\hat{f}_h(\mathbf{x})$, is a non-parametric estimate of f computed at point \mathbf{x} , defined as

$$\hat{f}_h(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n K_h(\mathbf{x} - \mathbf{x}_i) \quad (2.1)$$

where $K_h(d)$ is a circular kernel function that operates over all points at distance $d \leq h$ from \mathbf{x} , and h is called the bandwidth of the kernel (SILVERMAN, 1986; JONES; KAPPENMAN, 1992).

For any given point $\mathbf{x}_i \in X$ where $1 \leq i \leq n$, the KDE on dataset X is used to estimate the likelihood of a point \mathbf{x}_i being drawn from X . The probability estimated through the kernel density estimator may be interpreted as the “point density” at any $\mathbf{x}_i \in X$ (DEVROYE; KRZYŻAK, 1999; BHATTACHARJEE; GARG; MITRA, 2021).

A variety of kernel profiles may be applied for estimating the density using KDE. Even though the Gaussian kernel is one of the most frequently used kernel functions, other kernel profiles are used by the research community (SILVERMAN, 1986; MAFFEI et al., 2015a). Below we show the uniform, gaussian, and the inverted ones:

$$UK_h(d) = \begin{cases} a & , \text{ if } d \leq r \\ 0 & , \text{ otherwise} \end{cases} \quad (2.2)$$

$$GK_h(d) = \frac{1}{h\sqrt{2\pi}} \exp\left(-\frac{1}{2} \frac{d^2}{h^2}\right) \quad (2.3)$$

$$IK_h(d) = \begin{cases} h(b+c) - (\sqrt{h^2 - d^2})c & , \text{ if } d \leq h \\ 0 & , \text{ otherwise} \end{cases} \quad (2.4)$$

where d signifies the distance from a kernel centre \mathbf{x} to the target point \mathbf{x}_i , r is the radius of the kernel, and a is the height of the uniform kernel — typically $a = 1/(\pi r^2)$. The kernel bandwidth h is also known as the smoothing factor that controls the smoothness of the curve obtained from the KDE function. A higher value of h ensures a smoother curve of the kernel (BHATTACHARJEE; GARG; MITRA, 2021). For $IK(\cdot)$, b and c are respectively the height of the uniform circular kernel and the length of the semi-axis of the oblate ellipsoid (MAFFEI et al., 2015a). Figure 2.7 shows these three kernel profiles in their 2D and $2^{1/2}$ D shapes.

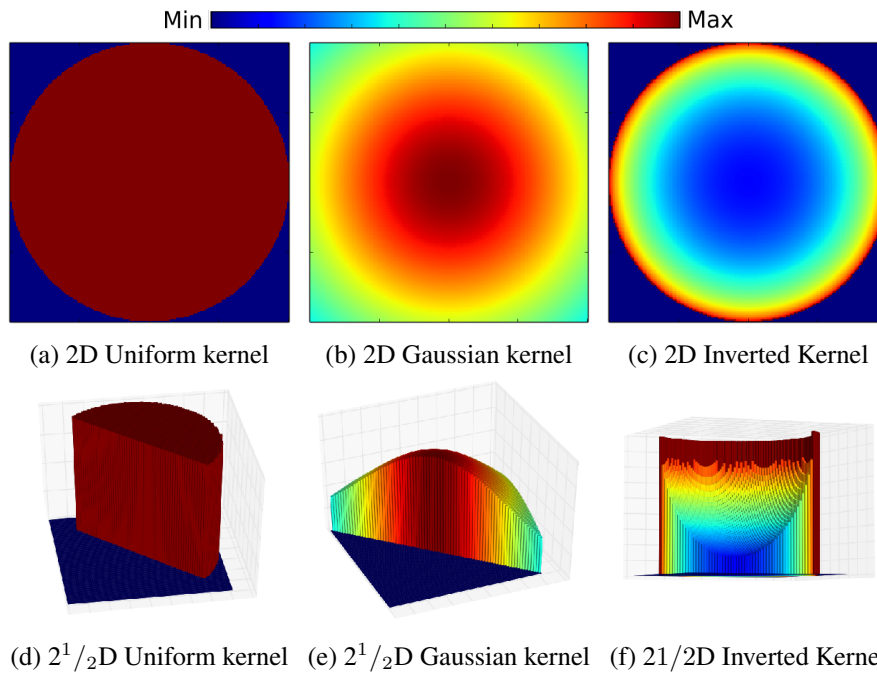


Figure 2.7 – Different kernel profiles shown in 2D and $2^{1/2}$ D. Adapted from (MAFFEI, 2017).

The KDE is a fundamental tool in statistics (DEVROYE, 1985; SILVERMAN, 1986; DEVROYE; LUGOSI, 2001; SCOTT, 2015) and machine learning (SCHÖLKOPF et al., 2002; GRETTON et al., 2012; MUANDET et al., 2017). However, it can also be used in many other fields, such as robotics, like in work proposed by Maffei et al. (2015a). The authors proposed an efficient KDE-based observation model for a localization approach in their work. The KDE is estimated considering the density of free space cells in a 2D grid map, as these cells are more abundant and less noisy than the obstacle cells. One big advantage of using KDE for robotic applications is that it is orientation independent and has a low computational cost (MAFFEI et al., 2015a). Their KDE of free

space surrounding a point \mathbf{x} is computed through an equation similar to Equation 2.1:

$$\hat{f}_h(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n s(\mathbf{x}) K_h(\mathbf{x} - \mathbf{x}_i) \quad (2.5)$$

where the function $s(\cdot)$ is defined as

$$s(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \text{ is inside a free space cell} \\ 0, & \text{otherwise.} \end{cases} \quad (2.6)$$

Figure 2.8 shows the Equation 2.5 with an uniform kernel applied to a 2D grid map. Given four map cells in four different spots of the map (1, 2, 3, and 4) indicated in Figure 2.8a, the circular KDE would consider the red cells highlighted in Figure 2.7a. The function $s(\cdot)$ in Equation 2.5 makes the KDE to consider only the free cells, as shown by the greenish cells in Figure 2.8c. The free space density estimation for each cell, estimated with KDE, is illustrated in Figure 2.8d. In this image, it is possible to see that the larger the number of free cells within the kernel area, e.g. spot number 3, the higher the density. Besides, this simple example also demonstrates that KDE is invariant to rotations. The KDE for spots 1 and 4 are equal, even though the area of free cells for each spot is in different orientations (horizontal and vertical). Point 3 has the highest KDE, which is explained by the highest number of free cells under the kernel's coverage among all the four kernels.

The idea of estimating the free space density with the aid of KDE was later used by Maffei and colleagues in a second work, Maffei et al. (2015b). This time, they were interested in using a robot to construct a topological map, to then localize itself using a laser range finder and odometry information. Their proposed algorithm used an observation model based on kernel density estimates, similarly to their previous work, Maffei et al. (2015a). The difference, however, is that this observation model separates the map into regions denominated words, classified based on the density of free space estimated with the KDE, number of observations, and segmentation orientation (MAFFEI et al., 2015b). The robustness of the free space density estimation in this second work motivated the authors to pursue a third work, Maffei et al. (2016). Now, Maffei et al. (2016) proposed a long-term place recognition, which extends their strategy that represents environment regions using words based on spatial density information extracted from laser readings. The novelty is in building multi-level words to account for possible changes in the observations of a place generated by non-static objects. For example, when a robot is in front

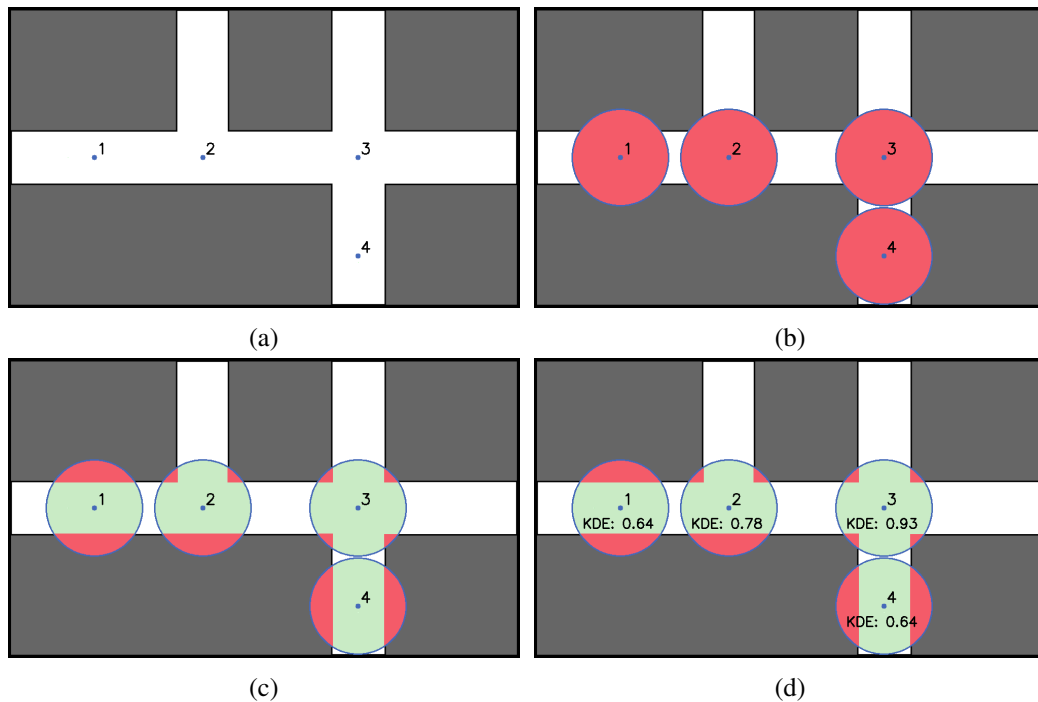


Figure 2.8 – Computing kernel density estimates in four different positions of a map. The gray cells represent the unknown region, whereas the white ones the free space. In (a) we have the four spots to compute the kernel density, in (b) the area of each kernel, in (c) the free cells within the area of the kernel, and in (d) the computed free space density.

of a door and estimates the free space density, the final KDE value will differ whether the door is open. For long-term operations, this may happen, and even though the density value is different, the region is still the same. In Chapter 4 we are using the KDE due to its low computational cost and efficiency, as part of a grid map segmentation approach.

2.3 Action in the environment

Making the right choices for the robot's action is crucial for many robotic applications. In the context of this thesis, the core of OS is making the proper decisions to bring the robot to the target position. Below we introduce an exploration technique we used in Chapter 4 to gain information about the environment, along with the presentation of the OS problem.

2.3.1 Exploration of environments

Exploration is one of the robotic problems that the research community has been working on for many decades (THRUN; BURGARD; FOX, 2006). Approaches that deal

with this problem come in handy when a robot is deployed in an environment where a map is unavailable. This scenario requires the robot to safely move around within the environment at the same time that it builds the map. This problem has two fundamental requirements: the safety of the robot while it is moving and the preference for movements towards unexplored regions of the environment (JORGE, 2017). A simple strategy to fulfil the safety requirement is to make the boundaries of obstacles to repel the robot. Similarly, the other requirement could be met by making the boundaries (frontiers) of unexplored regions attract the robot, as shown in Figure 2.9 (THRUN; BURGARD; FOX, 2006). Exploration approaches that are guided by the frontier of unexplored regions of the environment are commonly called frontier-guided (JORGE, 2017).

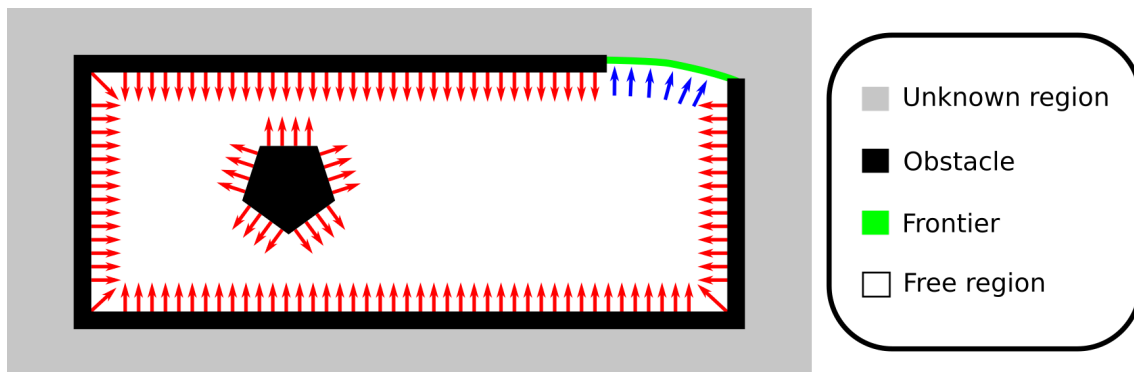


Figure 2.9 – Example of the two fundamental requirements of exploration at the boundaries of a partial map. The border of obstacles (black cells) has red arrows that represent the force that repels the robot. The border of unknown region (green cells) has green arrows that represent the force that attracts the robot. One way to update the potential field in the free space (white cells) is to consider a BVP, solving the Laplace equation considering the boundary-values in the green frontiers. Adapted from (JORGE, 2017).

Among the frontier-guided exploration approaches for exploratory tasks, we highlight the sequence of works proposed by Prestes and colleagues (PRESTES et al., 2002; SILVA et al., 2003; PRESTES; ENGEL, 2011). Initially, Prestes et al. (2002) proposed the use of a numeric solution to a boundary value problem (BVP) for the exploration problem. Their BVP exploration approach uses the gradient descent of the potential field generated over a grid to perform the map coverage with the robot. This potential field is computed with the finite difference method, by solving the Laplace equation,

$$\nabla^2 p(\mathbf{m}_i) = \sum_{j=1}^n \frac{\partial^2 p(\mathbf{m}_i)}{\partial x_j^2} = 0, \quad (2.7)$$

where $p(\mathbf{m}_i)$ is the potential value at position \mathbf{m}_i in the free space. In the grid map, the boundary of obstacles and unknown regions are considered Dirichlet boundary conditions.

The potential value associated to unknown cells is the minimum, i.e. 0, whereas the same value associated to obstacle cells is the maximum, i.e. 1. The free cells have their potential value updated with Equation 2.7 (PRESTES et al., 2002; SILVA et al., 2003). However, as they are working with a discrete regular grid map, they used the Gauss-Seidel method for the numerical approximation of Equation 2.7. Therefore, to update the potential value of free cells, they did

$$p(\mathbf{m}_i) = \frac{1}{4}(p(\mathbf{m}_l) + p(\mathbf{m}_r) + p(\mathbf{m}_u) + p(\mathbf{m}_b)) \quad (2.8)$$

where $p(\mathbf{m}_l)$, $p(\mathbf{m}_r)$, $p(\mathbf{m}_u)$, and $p(\mathbf{m}_b)$ are the left, right, top and bottom cells neighboring the center and reference cell $p(\mathbf{m}_i)$, respectively. Figure 2.10 illustrates an example of these cells.

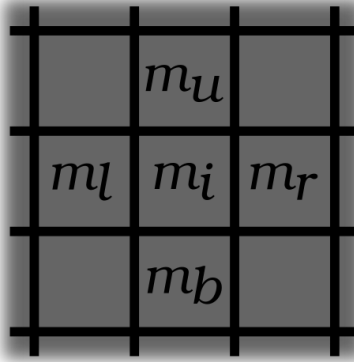


Figure 2.10 – Representation of part of a 2D grid map \mathbf{m} . The potential field computed in relation to a cell centered at \mathbf{m}_i , considers the four neighbors cells on its left, right, bottom and up, \mathbf{m}_l , \mathbf{m}_r , \mathbf{m}_b , and \mathbf{m}_u , respectively.

The Equation 2.8 applied to a 2D grid map to update the potential results in the exploratory behavior illustrated by Figure 2.11. The robot starts moving and it goes to the nearest frontier, Figure 2.11a. Then it goes inside a room and maps it, Figures 2.11b and 2.11c. As there is no frontier within the room, the potential field guides the robot towards other frontiers to keep exploring, as shown in Figure 2.11d. The BVP exploration approach uses the gradient descent of the potential field over a grid map to perform the map coverage. The angle of the robot's heading is computed based on the robot's current position on the potential field, i.e.

$$\theta = \arctan(p(\mathbf{m}_l) - p(\mathbf{m}_r), p(\mathbf{m}_u) - p(\mathbf{m}_b)) \quad (2.9)$$

where $\arctan(x, y)$ is the inverse tangent taken in the interval $[-\pi, \pi]$. The robot's speed can also be adjusted based on the potential field. The difference between the robot's ori-

entation and the negative potential gradient direction θ for the current position represents how much the robot has to turn. Hence, this difference can be correlated to the robot's speed, i.e., the larger the difference, the slower is the robot's speed (SILVA et al., 2003). Therefore, the BVP exploration has many advantages, including but not limited to smooth movements of the robot, easy understanding and implementation, and there are no local minima in the resulting potential field (MAFFEI et al., 2014; JORGE et al., 2015). As we mentioned in this subsection, the BVP is used in our work introduced in Chapter 4 to safely and smoothly move the robot to the most promising regions according to our search estimations.

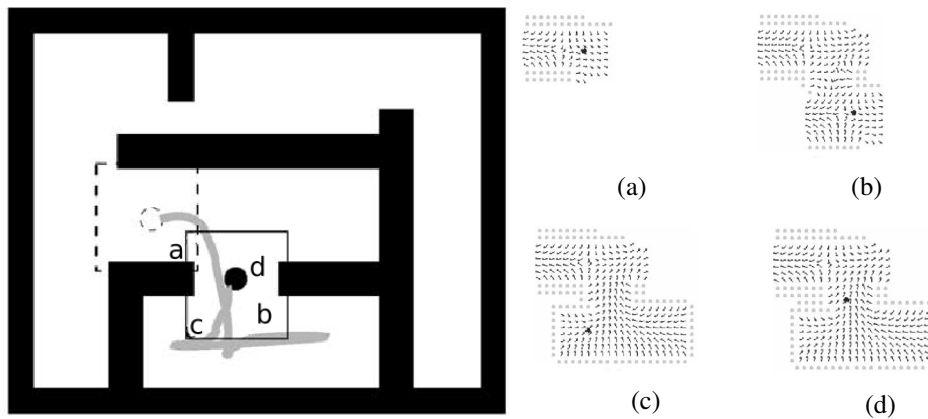


Figure 2.11 – Exploration process with the potential field. The trajectory followed by the robot is illustrated in the map, and four snapshots of the potential field during the exploration are shown in (a)-(d). Extracted from (PRESTES et al., 2002).

SR

2.3.2 OS problem

The OS problem relies on an efficient strategy for finding a target object in a large unknown indoor environment. Since our works presented in this thesis are based on a 2D grid map, the search strategies from these works reason over the map m . They estimate what cell m_i is currently more promising to find the target object while minimizing the total search cost. We define cost as the distance traveled by the SR during the search, as the longer the SR's path, the higher is the amount of resources (battery and time) it spends. The SRs we use throughout our experiments are equipped with a 2D LiDAR, to build a 2D grid map, RGB cameras, to gather visual cues for semantic information estimation, and RGB-D cameras to estimate the position of objects in relation to the robot's pose.

All sensors are fixed to the robot's body, and hence, we consider only the movements performed by a ground mobile robot.

Additionally, let $\varphi(\mathbf{m}_i)$ be the probability distribution for a map cell, \mathbf{m}_i , where the target object is in \mathbf{m} . Depending on the level of a priori knowledge of \mathbf{m} and $\varphi(\mathbf{m}_i)$, it is possible to address the OS problem in three different ways:

- **\mathbf{m} and $\varphi(\mathbf{m}_i)$ are known:** the problem becomes a sensor placement, aiming to reduce the search cost by moving the SR straight to the cell \mathbf{m}_i . Since the target object's position is already estimated, a path planning algorithm could be used to compute the shortest path between the SR's position, \mathbf{m}_r , and the target object's, \mathbf{m}_i .
- **only \mathbf{m} is known:** in case the map is available a priori (or acquired through a separate mapping step), the SR should either rely on a generic probability distribution or move through the environment to gather information. The inspection performed by the robot is to get information about the objects and update the probability distribution, and then estimate where the target object can be.
- **\mathbf{m} and $\varphi(\mathbf{m}_i)$ are unknown:** the robot needs to map the environment with the aid of a SLAM system, at the same time that it collects information to compute the probability distribution about the target object's position. Since the robot performs OS in an unknown environment, it has to tradeoff between expanding the mapped area and executing sensing actions to search for the target object carefully. This scenario is also known as the exploration vs. exploitation problem.

In this thesis, the last two points are addressed in different works, Chapters 4 and 5. In general, each of these works has an organisational semantic search strategy, i.e., it incorporates semantic information into the estimations to improve the performance. However, it is important to mention that these semantic search strategies consider common-sense knowledge, which is not environment-specific, and integrate high-level human concepts. In the context of this thesis, common-sense knowledge encodes semantic information inferred from numbers recognized in text signs and changes in the objects' position over time. Such information is valuable for our works because it reduces the amount of spots the robot has to search, and improves the search for a human-like performance. In Chapter 3, we review the main OS works proposed by the research community that have inspired this thesis. Besides, we also introduce more details about the OS problem.

3 RELATED WORK

The work in this thesis is closely related to two major research topics: OS approaches and spatio-temporal models of the environment. This chapter focuses on discussing the published works that aim to deal with these topics and that have been considered to develop this thesis.

3.1 Object Search approaches

The OS problem has been studied for many years in the robotics field. The proposed approaches range from multi-agent collaborating to search for an object (YE; TSOTSOS, 1996), to a single robot actively performing a semantic-based search (ZENG; RÖFER; JENKINS, 2020). After many years subtopics of research arose within the OS, such as Indirect and Active Visual searches. Despite this long period in which new approaches have been proposed, for the best of our knowledge, no detailed surveys in the literature shed light on this latter subtopic. However, it is possible to find comprehensive surveys on wider topics such as salient objects detection (BORJI et al., 2019), visual attention (BEGUM; KARRAY, 2010), and as pointed out by Aydemir et al. (AYDEMIR et al., 2013), active vision (CHEN; LI; KWOK, 2011; CHUNG; HOLLINGER; ISLER, 2011). It is important to mention that even though it has not been proposed as a survey, Aydemir et al. presented a comprehensive review of some of the most important works related to OS (AYDEMIR et al., 2013). Hence, we review other works not presented in (AYDEMIR et al., 2013), that are as important to the development of this thesis as the presented ones. This review allows us to show how this thesis compares to the visual OS body of research. It also shows how our contributions push the state-of-the-art further.

Previous works have investigated the OS task in numerous ways and based on different sorts of data. Ye and Tsotsos proposed one of the first works to deal with OS (YE; TSOTSOS, 1999), in which they provided a sensor planning system. They argued that the robot should change the sensing parameters to bring the target object into the camera's field of view. The proposed system was formulated as an optimization problem, i.e., maximize the probability of detecting the object with minimum overall cost. Hence, a robot equipped with a camera that could pan, tilt, and zoom, was used throughout their experiments. They decomposed the space of possible actions into a finite set to determine the sensing actions. The next action was selected based on comparing the likelihood of

detecting the target object and the action cost. They have successfully achieved their goal with better performance than those OS strategies with fixed action sequences.

In a series of papers, Aydemir and colleagues also explored the advantages of an adjustable visual sensor in the context of OS tasks (AYDEMIR et al., 2011; AYDEMIR et al., 2011a; AYDEMIR et al., 2011b; AYDEMIR et al., 2013). In Aydemir et al. (2011a), the authors proposed a spatial representation that consists of tree sub-representations. The first one is a 3D metric map used for obstacle avoidance, path planning, and viewpoint selection for object search. The second is a topological map, also called place map, that maintains the environment's topology. The last one is a conceptual map, which integrates all these other maps to infer the category of each place based on the room shape (elongated and square) and appearance (office-like, meeting room-like). Besides, the authors also proposed a planner for the OS based on the spatial relationships between objects and the environment (e.g., *table IN kitchen* or *book ON table*). First, it decides the overall search strategy based on the spatial representation (which objects should be found in which location). Then it computes a subset of all possible sensing actions that are most likely to bring the target object into the robot's field of view. In Aydemir et al. (2011b), it was used part of the contributions presented in Aydemir et al. (2011a). The spatial relation previously introduced was used here as the basis of their new strategy for an OS approach. They argue that the spatial relation is useful for OS tasks since it reduces the search space. If the OS system is aware of the relation *book ON table*, the search space reduces to the table area. The same applies for the case of *cup IN kitchen*, in which the search space is only the kitchen. Besides reusing the spatial relation concept, the authors also proposed the idea of grouping the spatial relations, i.e., *book ON table IN kitchen* in the case of the previous example. Their outcome was a strategy that can obtain near-optimal search behavior to find the target object. In Aydemir et al. (2011), it was proposed another OS approach, but for the first time using semantic spatial knowledge. Despite the hierarchical planner that is quite similar to the other works already present, the biggest novelty is the high-level conceptual and semantic information from the environment used on their OS approach. Semantic cues were used to guide the object search process, in which its semantic room category represents each discrete place from the environment. Due to the combination of low-level sensor percept and this high-level representation from semantic cues, the hierarchical planner efficiently performed the OS task. The advantages of using semantic information in OS tasks encouraged Aydemir et al. (2011) to propose another OS work. In Aydemir et al. (2013), the authors proposed

an OS approach for large unknown environments, and hence, the proposed system had to balance between exploring the environment to gain more information or perform the OS. Further, while exploring the environment, their approach guided the robot towards more promising unknown areas according to the robot's knowledge since its first movement. In terms of the strategy for searching the object, the authors proposed a planner that considers four actions: move, process view, calculate views, and search object. The first moves the robot to the desired place. The second one moves the robot to a viewpoint and runs an object detection algorithm on the image taken by the robot. The third calculates a set of viewpoints in a single room, aiming to point the camera towards the most promising objects within the room. The last one forms a subproblem for their planner whose the set of actions consists of move and process view to search for an object in a single room. For performing the presented actions, the planner also considers the semantic cues from the appearance, geometry, and topology of the environment and combines it with general semantic knowledge of indoor spaces to reason about locations of interest.

These works aforementioned were designed as a searching function that minimizes the search cost. Each action of moving either the robot or the camera has a cost, and the goal is to find the target object with the lowest total cost. In Aydemir and Jensfelt (2012), the authors fixed the robot's visual sensor, reducing the complexity of the searching function. The authors presented the 3D context idea: the correlation between local 3D structure and object placement in everyday scenes. They use the local 3D shape around objects as a signal of the placement of these objects. The advantage is that their approach can capture more complex 3D contexts without implementing specialized routines for the robot. Instead of looking for the object itself, they first find the places that are more likely to contain the object. Their results show that the local structure surrounding the target objects is a suitable indicator of object placement in scenes. Besides, their OS approach accurately predicted the location of the everyday objects included in the study. Lastly, the 3D context present in this work is machine learning-based, and hence, their OS approach has to be trained in every new environment to incorporate the singularities of the place. Therefore, the local information about how the objects relate to the 3D context must be known beforehand. Similarly to Aydemir and Jensfelt (2012) that made their RGB-D dataset publicly available, in (AYDEMIR; JENSFELT; FOLKESSON, 2012) Aydemir et al. also published a dataset called KTH. In this case, the dataset is composed of a set of floor plans that encompasses, in total, 37 buildings, 165 floors, and 6248 rooms. In addition to KTH, another contribution of their work was two methods for predicting indoor

topologies and room categories given a partial map of the environment. The goal was to predict what lies ahead in the topology of the environment through its topology.

The idea of relying on significant and visible landmarks to optimize the search was not used only in (AYDEMIR; JENSFELT, 2012). Zeng et al. exploited background knowledge about common spatial relationships between landmarks and target objects (ZENG; RÖFER; JENKINS, 2020). Their proposal, called Semantic Linking Maps (SLiM), maintained the belief over the locations of the target object and the landmark. Simultaneously, it accounted for probabilistic inter-object spatial relations. In contrast to the 3D context-based OS systems, Rasouli et al. proposed an attention-based OS system (RASOULI et al., 2020). They argued that an OS system must be responsive, directive, spatiotemporal, and efficient, which are the characteristics addressed by their model. It embedded visual attention in an n-step decision-making algorithm formalized as a 1st-order Markov process. The use of visual attention increased the robot's awareness of the environment. Hence, they used all relevant available visual information, leveraging the spatial and appearance information about the object. Rasouli and Tsotsos also relied on visual attention methods to reduce computational costs on their robotic visual search (RASOULI; TSOTSOS, 2017). They proposed a three-pronged probabilistic search algorithm that incorporated three forms of visual attention: viewpoint selection, saliency, and object-based models. On their model, attention is used to generate maps with highlighted areas in the image which are more likely to contain an object of interest. The experiments showed that the proposed three-tier attention framework decreased the search cost in terms of distance traveled, search time, and the number of actions taken. Saidi et al. explored a different robot than the other works that opted for wheeled robots since their OS system was proposed based on the specificities of a humanoid robot (SAIDI; STASSE; YOKOI, 2007). A visibility map, which constrains the sensor parameter space, was used to avoid unnecessary calls to the rating function that evaluates the interest of a potential next view through the analysis of the theoretical field of view.

The object's surroundings provide a significant benefit for OS approaches. In Chen and Lee (2013), it was proposed an OS approach for cluttered environments that are challenging scenarios due to the partial occlusion of objects by other ones. Some objects may be only half visible in such scenarios, and the authors' proposal used the object's surrounding and spatial constraints to aid the searching. As the authors argue, usually objects are neatly placed to fulfill many functional purposes, and hence, the searching space can be substantially reduced even before the start of the OS. The proposed approach works

in two phases. The first one is recognition, in which data is acquired to find out the number of objects in the scene and which map cells have a high probability of finding the objects. Both the object's 2D and 3D features are used for its recognition. The second one aims to generate an action for every candidate cell in the map and select the best one to be executed. Despite the promising ideas and results, Chen and Lee assume that the sizes, heights, locations, orientations, and accessible angles of all objects surrounding the target object are known in advance. Besides, they also assume these characteristics will not change throughout the searching process.

Sprute et al. (2017) proposed a complete system to support the elderly in their home environments. The system includes a service robot and a camera network to make the older person's home smart. The main proposal of this work is to use the camera network to benefit the service robot in performing the OS task, besides expanding the total area analyzed by visual sensors. The cameras in the environment reduce the searching space for the service robot and overcome the robot's sensor limitation. The hierarchical search system proposed by the authors consists of three layers: local search, global search, and exploration. The local search is activated when the object is within the robot's field of view. The global search is activated if the robot does not find the object locally, and here the environmental cameras are used. If these two first layers cannot locate the object, the robot explores the environment looking for it. Due to the substantial advantage of the camera network and the smart home integration with the service robot, the proposed system managed to find the objects within the environment during the experiments efficiently.

In Wang et al. (2018), the authors claim that if the robot behaves like a human in OS tasks, the searching efficiency and quality could be improved. Besides, they also argue that the semantic information of the entities in the environment could fill the gap between humans and the intelligent robot, so the robot could be trained by the typical human's knowledge to clarify the relations among entities. In light of this, they formulated the OS problem as a Partially Observable Markov Decision Process (POMDP), which is an idiomatic framework for modeling decision-making under uncertainty. The belief distribution of their custom POMDP was trained considering the semantic information of the room types and the objects. Besides the custom POMDP, the authors also proposed a graph structure called Belief Road Map (BRM), built along with the searching process in the unknown environment. The BRM is supposed to efficiently provide a path for the robot instead of using the whole grid map to estimate the path for the search.

It is worth mentioning that part of this thesis considers the exploration of unknown environments as part of the OS problem. We aim to perform OS in an entire unknown search space, which requires switching between the exploration of unknown regions and the exploitation in already known regions. Hence, it is important to discuss some works related to exploration. Here, they are divided into two significant groups regarding their goals. First, strategies that aim to explore the whole environment, usually finishing when the robot has visited the entire free area (LI et al., 2016; GIRDHAR; WHITNEY; DUDEK, 2014). Second, goal-directed strategies that aim to reach a goal, such as searching for an object, a room, or a person.

Some of these exploration-based OS approaches use a semantic map, whereas others use semantic properties from objects. The framework proposed by Veiga et al. (VEIGA et al., 2016) searches for objects in domestic environments. It is composed of a system that perceives the query object in RGB-D images through inference and sensor information. The outcome of this process, called knowledge, is saved and updated in a semantic map. Experiments in a realistic apartment have shown that their framework worked well in practice, presenting a reliable and efficient search approach.

Another significant work that searches objects in domestic scenarios is Rogers' et al. (ROGERS; CHRISTENSEN, 2013). In contrast to (VEIGA et al., 2016) that proposed a modular system, their approach considered the context of the environment. A graph, connecting places and objects within these places, is used to predict objects' presence (or absence) based on the room categories. The reasoning over the graph, combined with a planner, is used to perform an object search task. Experiments showed that the robot was able to find objects in the environment.

Talbot et al. (TALBOT et al., 2016) and Schulz et al. (SCHULZ et al., 2015) proposed navigation approaches that are also goal-directed, despite not being exploration ones. The idea of an original and abstract map that links symbolic spatial information with observed symbolic information and actual places in the real world was firstly introduced by (SCHULZ et al., 2015). This map is used to make inferences about the location of places. Later, Talbot et al. (TALBOT et al., 2016) extended the idea of the abstract maps, proposing a novel method that defines the topological structure and spatial layout information encoded in spatial language phrases. The system has shown to complete the task by traveling slightly further than the optimal path.

Despite the good outcomes from the solutions presented by the papers mentioned above, there is still room for improvements. In (AYDEMIR et al., 2013) Aydemir et al.

depended on prior semantic knowledge about indoor spaces obtained from databases. Talbot et al. (TALBOT et al., 2016) and Schulz et al. (SCHULZ et al., 2015) depended on a priori abstract maps. Veiga et al. (VEIGA et al., 2016) required beforehand information to learn about objects and the environment. Additionally, it used a 3D recognition-based framework from the Point Cloud Library (PCL) for object recognition, which is computationally expensive. Rogers et al. (ROGERS; CHRISTENSEN, 2013) also implemented PCL to segment data from RGB-D sensor, continuing to cluster the points, which is a heavy workload for computers. It is also important to highlight that none of them has explored the benefits of textual information available in the environment.

In contrast to the works that rely on semantic information to improve the robot's performance in OS tasks, Rasouli et al. (2020) proposed a system that guides the robot towards the target object using the relevant stimuli provided by the robot's visual sensor. Visual attention techniques are used to extract visual information from the environment actively. In combination with a non-myopic decision-making algorithm, the author's proposal leads the robot to search more relevant areas of the environment to find the target object. The results indicate that visual attention improves the searching process, but it also depends on the nature of the OS task and the complexity of the environment.

3.2 Spatio-Temporal models

Most of the works proposed by the research community in Mobile Robotics ignore or filter the changes in the environment since they are considered noise and only disturb the estimations. The idea of modeling and incorporating the environmental changes into different robotic solutions is relatively new. Below we present the works based on spatial-temporal information, which most inspired this chapter work in terms of modeling the environment changes.

In Krajník et al. (2014), it was proposed a topological localization approach for service robots in dynamic indoor environments. It explicitly uses information about environment changes by learning and modeling the spatio-temporal dynamics of the robot's area. First, the robot learns the changes in the surroundings of each pre-defined location over one week, and it models the changes using the proposed spectral representation. Then, when the robot estimates its position within the map, it tries to match its current observation to the predicted representations of each location's surroundings for that specific time. According to the authors, the proposed localization approach can predict environ-

mental changes in time, allowing the robot's localization improvement during long-term operations in populated environments. However, the authors assume that the environment's appearance is affected by a set of hidden, periodic processes in mid- to long-term perspective, and that the environment's dynamics can be described by the frequency, amplitude, and time shift of these processes.

Besides the localization field, the time and environment changes were also considered within the human-robot interaction field. Vintr et al. (2019) introduced a spatio-temporal representation for service robots to anticipate the human presence in human-populated environments. Their proposal aims to model periodic and temporal patterns of people's presence, considering their routines and habits. The proposed representation projects the time onto a set of wrapped dimensions representing the periodicities of people's presence, and hence, it can make long-term predictions of human presence. These predictions allow service robots to schedule their tasks in a more suitable way not to bother humans.

Krajník et al. (2020) explored the Chronorobotics, a new area introduced by them, that studies the experiences that autonomous systems can gather when observing human-populated environments for an extended period of time. The goal in Chronorobotics is to provide robots capable of adapting to naturally cyclic dynamics of the human-populated environments. Their work proposed methods that introduce the notion of dynamics into spatial environment models, which end up in representations that provide service robots the ability to anticipate future states of changing environments.

Narayana, Kolling and Fong use a semantic map as the user-facing interface on their fleet of floor-cleaning robots (NARAYANA et al., 2020). They argue that these robots may operate in dynamic environments, and hence, it is necessary to enable the mapping to lifelong applications. They update their map based on the changes in the environment, keeping only the most updated version of the semantic map, overwriting the past versions. They use algorithms to detect when there are conflicts in the update of the semantic map, and to resolve them with an additional map layers called map-meta with additional meta-semantics. Their semantic map represents classes such as walls, rooms and doors.

The last work discussed in this section is another work proposed by Krajník et al. (2015). The authors argue that in human-populated environments, the object locations are impacted by human activities that tend to exhibit daily and weekly periodicities. Hence, identifying and modeling these periodicities generates a more accurate representation of

possible object locations, thus reducing the search space. Their search is formulated as a path planning problem in partially known environments, in which the probability of object occurrences at particular regions is a function of time. A traditional topological map represents the probability of object locations. Each node is associated with a temporal model that represents the dynamics of the object occurrence at the particular location. The experiments in different datasets show that explicit representation of the long-term periodicities of environment dynamics speed up the search process due to the search space reduction. Despite the promising results, the work has two assumptions. First, the topology of the environment where the robot is operating has to be known in advance. Second, the target object locations are influenced by human activities that exhibit a certain degree of periodicity.

3.3 Semantic map and heat map

Despite not proposing an OS system, Pangercic et al. introduced an representation and acquisition of Semantic Objects Maps (SOM⁺), which provides information for autonomous SRs (PANGERCIC et al., 2012). Their SOM⁺ represent all the furniture entities of kitchen environments including cupboards, electrical devices, tables, counters, positions, appearances, and articulation models. In addition to the objects' pose, it also includes the appearance and articulation of furniture objects, so the robot can perform fetch and place tasks more efficiently. Lastly, the area of operation limits to kitchens, where the robot can manipulate drawers, doors, cupboards, and other objects.

A second non-OS work that it is worth to mention here is the one proposed by Oksanen et al. (OKSANEN et al., 2015). They use a heat map to highlight the most popular places to do sports in a given region, and it was applied to public cycling workouts to represent the density of the trajectories and the diversity of the users. They only register the trajectory that has been taken by the cyclist, not matter when (day or hour) the user was working out.

3.4 A discussion on the Related Works

There are several aspects of our thesis that push it beyond the current state of the art, summarized in Table 3.1. Compared to most of the OS works discussed within this

section, ours considers the organisation of the environment as search cues and does not ignore the environmental changes over a period of time. The works proposed by Krajník and colleagues have shown that it is possible to model the environment changes, and spatial-temporal-based approaches present considerable improvements and robustness. What sets our thesis aside from Krajník et al. (2015), for example, is that we do not assume that the object's location exhibit a certain degree of periodicity. Besides, it is not necessary to collect data for an extended period to then start the search. Instead, our proposals do not require any pattern or periodicity. Besides, they do not require any data beforehand, which helps their deployment in practice. This is a considerable advantage compared to the works that demand the environment's map, object-object relation (or object-place relation), or the information about the target object's geometry or color.

Table 3.1 – Table comparing OS and spatial-temporal works.

	Object Search	Environment map in advance	Object location knowledge	Object-object relation	Object-place relation	Spatio-temporal	Object knowledge	Periodicity dependence	Long-term application
(YE; TSOTSOS, 1999)	✓	-	-	-	-	-	-	-	-
(AYDEMIR et al., 2011a)	✓	-	-	-	✓	-	-	-	-
(AYDEMIR et al., 2011)	✓	-	-	-	✓	-	-	-	-
(AYDEMIR et al., 2011b)	✓	✓	-	✓	✓	-	✓	-	-
(AYDEMIR; JENSFELT, 2012)	✓	-	-	-	✓	-	✓	-	-
(AYDEMIR et al., 2013)	✓	-	-	-	✓	-	-	-	-
(CHEN; LEE, 2013)	✓	-	-	-	✓	-	-	-	-
(SPRUTE et al., 2017)	✓	-	-	-	-	-	-	-	-
(WANG et al., 2018)	✓	-	-	-	✓	-	-	-	-
(RASOULI et al., 2020)	✓	-	-	-	-	-	-	-	-
(KRAJNÍK et al., 2014)	-	✓	-	-	-	✓	✓	✓	✓
(VINTR et al., 2019)	✓	-	-	-	-	✓	-	✓	✓
(KRAJNÍK et al., 2020)	-	-	-	-	-	✓	-	✓	✓
(KRAJNÍK et al., 2015)	✓	✓	-	-	-	✓	-	✓	✓
Ours	✓	-	-	-	-	✓	-	-	✓

4 NSOS: USING NUMBERS TO INTERPRET THE ORGANISATION OF UNKNOWN ENVIRONMENTS

The first of our contribution is called Number-based Semantic OS system (NSOS), and its goal is to guide an SR through an unknown environment until it finds a target door sign. The novelty of our system is that it relies on detecting numbers from door signs to estimate the organisation of indoor environments, aiming to reduce the search cost. Besides, another aspect of NSOS is the combination of semantic and geometric clues to improve the search estimations. The former clue type is inferred from the numbers of several door signs found in the environment, and they are useful for understanding how the doors are organised. The latter type is computed from the geometry of the environment, e.g., the corridors and intersections, and it indicates which regions are more likely to finding doors. An example of using our NSOS could be an autonomous courier SR that delivers packages to an older person in a specific apartment within a large-scale and unknown building, where a previous map is not available.

This section details the basic modules that compose our NSOS system. It starts with the proposal and contributions in Section 4.1, followed by the explanation of the basis for our NSOS system in Section 4.2. It details the Mapping module in Section 4.2.1, to explain how the 2D grid map is built. The Map Segmentation module is introduced in Section 4.2.2, which presents how the map is split into different segments. Section 4.2.3 describes how the numbers are extracted from the door signs through computer vision algorithms. The planner of our NSOS system is described in Section 4.3. Finally, this chapter is finished with the experiments in Section 4.4, and the summary in Section 4.5.

4.1 Proposal and contributions

Our NSOS efficiently searches for a target object that is represented by the number of door signs, called *goal-door*, in large-scale and unknown buildings. Different OS approaches have been proposed by the research community as earlier presented in Chapter 3. In many of them, the robot is tasked with finding objects based on their visual appearance or position in the environment, e.g. an yellow mug on the kitchen table. However, to the best of our knowledge, OS systems that search for door signs (not areas in general) and take numbers (in text format) as input to their strategy are not well explored yet.

The novelties presented in this chapter in comparison to the already published systems are threefold:

- **Number from door sign as target object:** The target of our proposed semantic OS system, NSOS, is a specific door sign within an unknown indoor environment. Several other OS systems look for objects that are part of our daily life, such as kitchen utensils or office supplies. However, in some cases finding a door sign is as important as finding the objects from the previous example. The importance is because the door sign is usually associated to a room of the environment, where the SR could perform other tasks afterwards. This novelty enables, for example, autonomous courier SR to perform the final part of the delivery task, which happens after it arrives at the buildings, where there is no map available. Our system is relevant in moments such as the COVID-19 pandemic or any other outbreak. In these situations, people are recommended to avoid social contact, which forces them to stay at home. Hence, the use of fully autonomous SRs for performing the entire delivery task, from the restaurant straight to the customer's door, saves people of being exposed to the threat, mainly the elderly people. The Starship Technologies company has already deployed their fleet of courier SRs, supporting the importance of robots in food and package deliveries. Another example of how our NSOS could be used will be presented in Chapter 6.
- **Organisational semantic information inferred from numbers:** Our NSOS system relies on information inferred from numbers as a visual cue, and more specific numbers extracted from the door signs. Large buildings, for instance, are divided into many small rooms, and usually comply with a pattern of signing each room (AYDEMIR; JENSFELT; FOLKESSON, 2012; AMHERST, 2012; HAINES, 2014; UNIVERSITY, 2017). Different from depending on color, visual features, or 3D templates, to perform the search, using numbers allows our NSOS to infer high-level information about the environment. Such information may suggest how the environment and the door signs are organised, which is useful during the search. If we could analyze humans behavior while looking for a door sign in an unknown environment, most of them would try to figure out the door signing arrangement. They would avoid exploring the entire building by analyzing how the door signs are related to each other to infer whether the current corridor is promising in terms of containing the goal-door. A courier SR could behave in a similar way to efficiently perform the searching task. Hence, our NSOS system processes the numbers to in-

fer semantic information from them. The inferred information is the odd and even characteristics and whether the sequence of numbers is increasing or decreasing. In addition to the semantic information, our NSOS system also considers geometric information, which is the distance between the estimated robot's pose and the unknown regions, and the history of the robot's orientation.

- **Organisational semantic information to complement the robot's geometric perception:** we also present an analysis of the advantage of using numbers from the door signs and the inferred semantic information as input to our NSOS system. This way, it is not limited to only geometric information. The combination of this semantic information, inferred from the numbers, and the geometric information, extracted from the environment, are useful inputs for the reasoning of our system. It permits the efficient computation of search steps, guiding the robot towards areas more likely to lead to the target room. Lastly, in addition to this computation, which is fully probabilistic, our system also builds a 2D map of the environment. It is segmented based on KDE, introduced in Section 2.2.3, according to different densities of free space (MAFFEI et al., 2015b). Our system considers the laser readings to build the 2D map, and the images of an RGB camera to extract the numbers from the door signs, and the other types of character are not used for now. The numbers are used to infer semantic information and search cues, whereas the NSOS system indicates which direction is more likely to contain the goal-door to guide the SR.

4.2 Basis for NSOS system

Our proposed system performs OS, i.e., it guides the SR through an unknown environment until the robot reaches a specific location that contains the target object. Our system focuses on indoor environments, such as buildings with many rooms identified by door signs. One of its advantages is that it does not require any a-priori knowledge about the environment, such as the door signs arrangement or where the SR should be heading. The target object to be found is a door sign, which is identified by a number.

Our system comprises four modules: Mapping, Image Processing, Map Segmentation, and Semantic Planner. The Semantic Planner module, presented in Section 4.3, is the main contribution of this chapter. It requires a base system to work, composed by the first three other modules, that are discussed in Sections 4.2.1, 4.2.2, 4.2.3. The first of these three modules, Mapping, aims to build a 2D grid map of the environment. For

our NSOS system, a 3D map would not be significantly beneficial, and the extra computational cost to build and update a 3D map does not worth it. The next module, Image Processing, processes the images taken by two RGB cameras, and it analyses them to recognize the number from door signs. Once identified, the module updates the 2D grid map to include the numbers at their respective side. For instance, if the robot’s left camera captures an image that contains a door sign, such number is drawn in the map, in the left side of the current robot’s position. The same applies for the robot’s right camera. Figure 4.1a illustrates this process. The third module, Map Segmentation, is responsible for segmenting the free space of the 2D grid map according to its size using the Kernel Density Estimation (KDE) approach introduced by Maffei *et al.* (MAFFEI *et al.*, 2015b). This module also assigns to each segment of a corridor its respective list of door signs, which is the list of numbers recognized while the SR was within the corridor.

4.2.1 Mapping module

The Mapping module considers that an SR is equipped with a 2D LiDAR sensor. The sensor measurements are used to build a 2D occupancy grid map m of the environment, like shown in Figure 4.1a. In simulation, the grid map m is built using the Histogramic In-Motion Mapping (HIMM) technique (BORENSTEIN; KOREN, 1991), that takes as input the LiDAR sensor readings. We assume there is no error in the robot motion, as we explained in Section 2.1, but this assumption holds true only for the simulation experiments. When carrying out the experiments with the physical robot, we used the SLAM system called GMapping (GRISSETTI; STACHNISS; BURGARD, 2007b).

Over m , the Mapping module also computes the Voronoi diagram to have the center cells of the free spaces, represented by the green lines in Figure 4.1b. The yellow region represents the free space visited by a circular kernel centered at the robot’s position, i.e., the circle centered at the robot’s pose. Based on that, the BVP smoothly moves the SR through the environment, avoiding obstacles and keeping it as close as possible to the Voronoi cells.

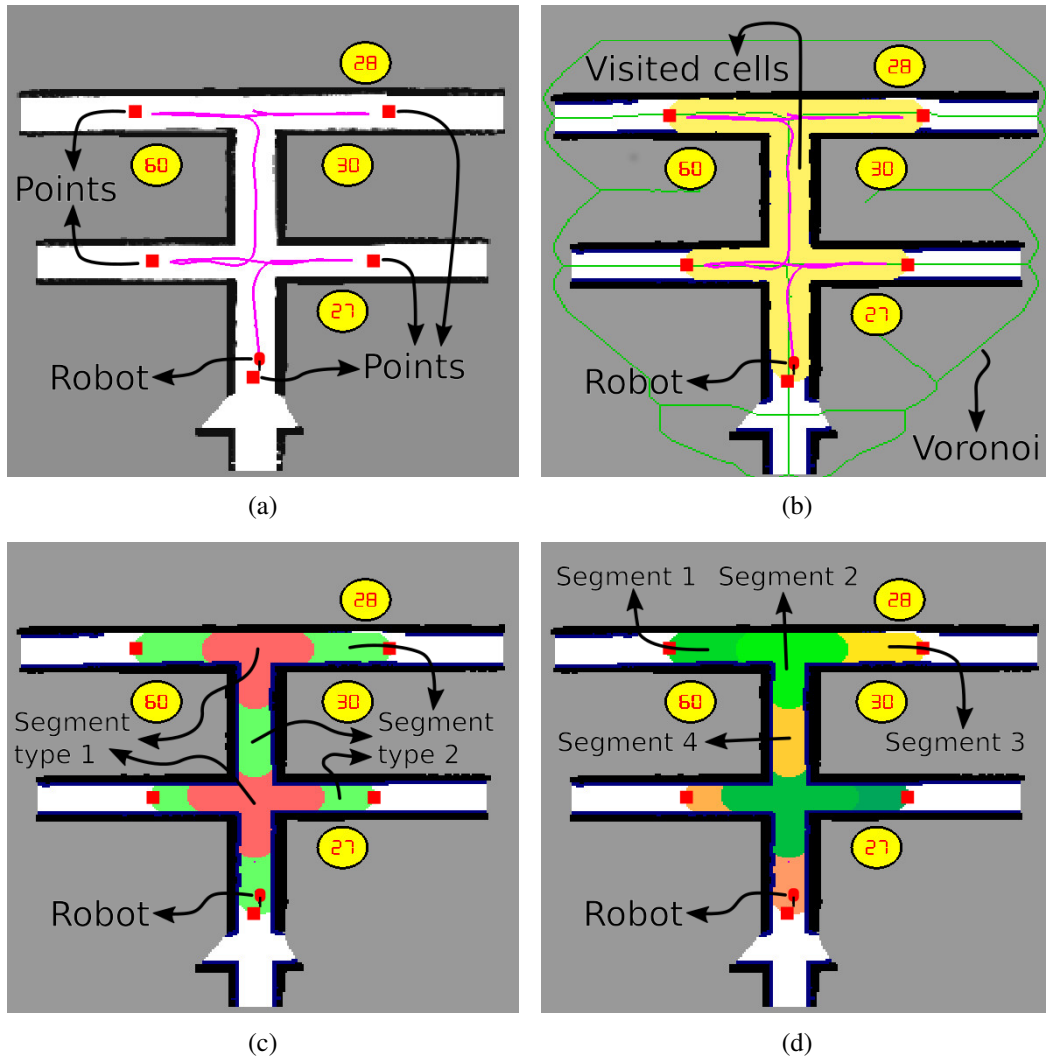


Figure 4.1 – Example of our mapping and segmentation modules. (a) It shows the robot, the white area representing free cells, the pink line as the robot path, and the five frontier points, (b) the marked yellow areas representing the visited cells and the Voronoi through the green lines, (c) the segmented map, with the visited cells from (b) segmented into two different segment types, and (d) the segment identification using different colors with four distinct segments indicated.

4.2.2 Map Segmentation module

The Segmentation and Mapping modules are executed simultaneously, aiming to split the free space of m into multiple regions according to the density of free space. Every segmented region, called a segment s , is associated to a group of cells m_i of the grid map m . In m , there might be different types of segments. Figure 4.1c illustrates the case in which the Segmentation module considers only two types. In this case, all green segments, or all the red ones, have a similar density of free space computed using the KDE. Besides their type, each segment is also singularly identified, such as s_j for the j -th segment in m . Figure 4.1d shows the segmented 2D map as if each segment was

identified as one color, to highlight that they are identified differently (even though there are only two segment types in such figure).

KDE computes the estimation of a given region surrounding a cell \mathbf{m}_0 , $\Psi(\mathbf{m}_0)$, which is explained in Section 2.2.3, centered on \mathbf{m}_0 by

$$\Psi(\mathbf{m}_0) = \frac{1}{|T|} \sum_{\mathbf{m}_i \in T} Q(\mathbf{m}_i, \mathbf{m}_0) UK(\|\mathbf{m}_i - \mathbf{m}_0\|). \quad (4.1)$$

where $K(\cdot)$ is a uniform circular kernel function that over all cells at distance $d \leq r$ from \mathbf{m}_0 , defined as

$$UK(d) = \begin{cases} 1 & , \text{if } d \leq r \\ 0 & , \text{otherwise,} \end{cases} \quad (4.2)$$

where r is the radius and d is the Manhattan distance from the current cell being measured, $\mathbf{m}_i \in T$, to the centre of the kernel, cell $\mathbf{m}_0 \in \mathbf{m}$. $T \in \mathbf{m}$ is a subset of cells that are at maximum r farther from \mathbf{m}_0 . The function $Q(\mathbf{m}_i, \mathbf{m}_0)$ tests whether a cell \mathbf{m}_i is free, and it is defined as

$$Q(\mathbf{m}_i, \mathbf{m}_0) = \begin{cases} 1 & , \text{if } \mathbf{m}_i \text{ is a cell that belongs to the free space region connected to } \mathbf{m}_0 \\ 0 & , \text{otherwise.} \end{cases} \quad (4.3)$$

According to Equation 4.3, $\Psi(\cdot)$ returns different density estimations based on the number of obstacle or unknown cells are surrounding \mathbf{m}_0 . Assuming that the Segmentation module considers different values, and given that Equation 4.1 estimates the density of free space surrounding a cell $\mathbf{m}_0 \in \mathbf{m}$, $\Psi(\cdot)$ can be used in the Segmentation module as

$$\Upsilon(\mathbf{m}_0) = \left\lfloor \frac{\Psi(\mathbf{m}_0)}{\delta} \right\rfloor \quad (4.4)$$

where δ is a threshold that defines how many different densities of free regions are considered by the segmentation function, $\Upsilon(\cdot)$. Therefore, a high δ means Equation 4.4 considers few different densities, whereas a low δ is the opposite.

A segment s represents a group of free and adjacent cells from \mathbf{m} that have the same $\Upsilon(\cdot)$. Figure 4.1d demonstrates different segments, in which each one has a different color. For example, given \mathbf{m}_0 and \mathbf{m}_1 as two free and neighbouring cells in \mathbf{m} , and that $\Upsilon(\mathbf{m}_0) = \Upsilon(\mathbf{m}_1)$, then both belong to the same segment s_0 . Otherwise, a new segment s_1 is created and \mathbf{m}_1 is associated to it. Thus, the segmentation of free adjacent cells from

m is based on Equation 4.4.

4.2.3 Image Processing module

The last module that completes the basis of our NSOS system is the Image Processing. It aims to recognize the number of door signs that may be in an RGB image. The idea here is to use one existing character recognition algorithm (ZHANG et al., 2013; JUNG; KIM; JAIN, 2004; NEUMANN; MATAS, 2012), since this is not the focus of this thesis and any approach can be used. The chosen work is the one proposed by Neumann and Matas (2012) due to its real-time recognition aspect and its robustness against noise, and low contrast of characters. Besides, it does not require any information or preparation beforehand. The work proposed by Neumann and Matas (2012) is presented in details in Section 2.2.1.

For a given image I that was captured by the robot at cell m_r , e.g., Figures 4.2a and 4.2c, this module returns a list, L , containing the recognized number from door signs. In the case of Figure 4.2, L would contain only the number 228. Figure 4.2 also shows where the detected door signs are included into the 2D map m . Figure 4.2c shows an image taken by the robot's right camera, and hence, the number is included in m , at the right side of the robot's position, m_r , as shown in Figure 4.2b. Given that the goal of signing rooms is to provide a unique door sign for each of them, it is assumed that there are not two door signs in an environment identified by the same number. In regions where there are multiple door signs in sequence with the same number, our system will consider only one instance of all these door signs. After receiving L , it must be merged with the numbers of the door signs from the segment that m_r is associated. For this process, it is important to define $S(m_i)$ as a function that returns the segment in which the cell m_i is associated with. In addition, there is also the function $L(\cdot)$ that returns the list of door signs from a segment. Thus, each door number $l \in L$ is included in the list of door signs from the segment of m_r , $l \cup L(S(m_r))$. Besides, each l has an occurrence number that increases by one every time the image processing algorithm recognizes it. If the robot revisits a place and recognises a l that already exists in $L(S(m_r))$, then its occurrence number is summed to the one in $L(S(m_r))$.

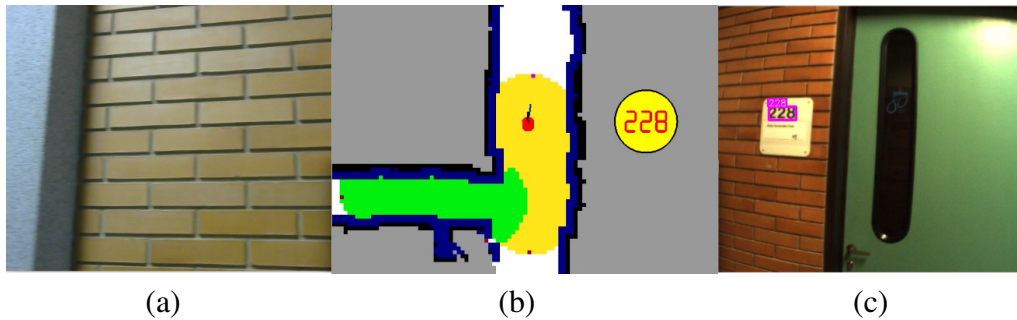


Figure 4.2 – Example of the Image Processing module processing two images, in which (a) is an image taken from the left camera, and (c) an image taken from the right camera. (b) shows the 2D map of the environment and the position of the door sign number 228.

4.3 Semantic Planner of our NSOS system

The previous Section 4.2 explained the necessary components that compose the basis of our NSOS system, i.e., the Mapping, Segmentation, and Image Processing modules. The explanation continues with the Semantic Planner module, presenting how the planner decides whether the SR should continue its search in the current region to find the target, or change its path to a known region. To the rest of this section, imagine that an SR has partially visited the environment while running the necessary components of our NSOS system. Then, there will be a partial map, segmented, and with all the so far recognised doors included. This section presents the details of our planner assuming the existence of such a map, aimed to help the explanation.

Our semantic planner is composed of five different parts, in which two of them are semantic-based, Growing Direction and Parity, and the other three are geometric-based, Doors and Robot Orientations, and Closeness. Combining them leads to a planner that is neither exclusively semantic nor geometric. This non-exclusivity characteristic is suitable for situations where the environment does not have semantic cues to be considered by our system. All the five factors are presented individually in this section, introducing the semantic-based first and then the geometric-based ones. However, as this section follows a top-down fashion to introduce the whole planner, the computation of the estimation about where the SR should go, which combines all the five factors, is presented before them. Therefore, the reader can have a general idea of how the factors are used and later understand how they work.

4.3.1 Combining the geometric and semantic factors to estimate where to go

Our NSOS system analyses the environment during the searching process while the SR has not found the goal-door. If it realizes the current region of the environment is not promising, the system guides the SR to another direction. To decide the best frontier to go given the set of frontiers, for each candidate cell $\mathbf{m}_i \in \mathbf{C}$, the planner calculates its attractiveness factor $\varphi(\mathbf{m}_i)$. This factor is the outcome of the combination of five factors briefly presented earlier. These candidate cells in \mathbf{C} are the ones in the center of the free space, i.e., in the Voronoi, and within a frontier, which is the boundary between visited and not visited cells. Graphically speaking, five candidate cells are shown in Figure 4.1a, represented by the red dots near the pink line ends. In this case, $\mathbf{C} = \{\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_5\}$. The visited cells are the free cells that were within \mathbf{T} , represented by the yellow region in Figure 4.1b, whereas the white region represents the free cells that is not close enough to the kernel centered at the robot's position. The explanation here is divided into two parts, in which the semantic factor is presented before the geometric factor. In the following subsections, the components of each factor, i.e., Growing Direction, Parity, Robot and Door Orientations, and Closeness, are presented.

The semantic factor, $\text{SF}(\mathbf{m}_i)$, combines the Growing Direction and the Parity factors, $\varphi_g(\cdot)$ and $\varphi_p(\cdot)$, respectively. The idea of the first factor is to return high values when the segment $S(\mathbf{m}_i)$ is more likely to contain the goal-door given the door sign sequence. On the other hand, the second part of $\text{SF}(\mathbf{m}_i)$ aims to analyze the parity of $S(\mathbf{m}_i)$ and compare it to the goal-door parity. When an SR is in a $S(\mathbf{m}_i)$ that is not likely to contain the goal-door, either due to $\varphi_g(\cdot)$ or $\varphi_p(\cdot)$, it should go to another path and continue the active search. Both Growing Direction and Parity factors are essential for our NSOS. If the robot is moving through a corridor where the numbers from the door signs have their parity different from the goal-door, there is no reason to continue the search there. Similarly, if the numbers of a sequence of door signs is increasing (or decreasing) away from the goal door, searching in that region is pointless. Hence, if either of these factors indicate the corridor is not promising, the robot should continue the search in another region, regardless the other factor. Then, in $\text{SF}(\mathbf{m}_i)$ they are multiplied by each other. If one is low, the result of $\text{SF}(\mathbf{m}_i)$ will end up being low as well, even when the other is high. It is important to highlight that $\text{SF}(\mathbf{m}_i)$ is completely probabilistic, and considering to how both $\varphi_g(\cdot)$ and $\varphi_p(\cdot)$ are modelled, $\text{SF}(\mathbf{m}_i)$ becomes robust to outliers. The Semantic

factor is given by

$$\text{SF}(\mathbf{m}_i) = \varphi_g(\mathbf{m}_i)\varphi_p(\mathbf{m}_i) \quad (4.5)$$

Differently, the geometric factor, $\text{GF}(\mathbf{m}_i)$ combines the three other factors (Door Orientation, Robot Orientation, and Closeness), which are computed based on the geometry of the environment. During the search, it is important that the semantic planner makes the most optimal decisions, like when the SR is in a crossroads and there are multiple options of where to go. In situations like this, the Door Orientation factor aims to suggest which corridors are more likely to contain door signs. Based on the robot's orientation when it recognised the all past door signs, this factor estimates the best option. Similarly, the Robot Orientation factor estimates how the corridors of the environment are organised. It projects which corridors are more frequent to happend based on their orientation (e.g. vertical or horizontal). These two orientation-based factors are important to the semantic planner because they provide vital geometric information of the organisation of the building, e.g., in which corridors is expected to contain door signs. The goal of the last factor, Closeness, is to save SR's battery. Among the segments the planner has to choose to take the robot, it measures how close each one is to the robot's current position. The geometric factor multiplies the Robot and the Door Orientation factors, $\varphi_r(\cdot)$, $\varphi_o(\cdot)$ respectively, by the Closeness one, $\varphi_c(\cdot)$. The idea is that the further the segment is, the less it matters. Then, the outcome of these multiplications is summed to the $\varphi_c(\cdot)$. The geometric factor is given by

$$\text{GF}(\mathbf{m}_i) = \frac{(\varphi_o(\mathbf{m}_i) + \varphi_r(\mathbf{m}_i))\varphi_c(\mathbf{m}_i) + \varphi_c(\mathbf{m}_i)}{3.0}. \quad (4.6)$$

Finally, in order to compute the best $\mathbf{m}_i \in \mathbf{C}$, i.e. the \mathbf{m}_i that is more likely to contain the goal-door, each of them is submitted to the following equation

$$\varphi(\mathbf{m}_i) = \text{SF}(\mathbf{m}_i)\alpha + \text{GF}(\mathbf{m}_i)(1.0 - \alpha). \quad (4.7)$$

Here, α is a threshold that controls the importance of the $\text{SF}(\mathbf{m}_i)$ and $\text{GF}(\mathbf{m}_i)$, and it ranges as $0 \leq \alpha \leq 1$. To estimate the candidade cell, \mathbf{m}_i^* , in which its $S(\mathbf{m}_i)$ is more likely to contain the goal-door, we do

$$\mathbf{m}_i^* = \arg \max_{\mathbf{m}_i \in \mathbf{C}} (\varphi(\mathbf{m}_i)). \quad (4.8)$$

4.3.2 Growing Direction factor

Usually, doors sign of buildings are arranged in sequence and sorted (either in increasing or decreasing order). For example, the number of the first door sign in a corridor is smaller than the last one, or in the other way around. This characteristic can be inferred through the door sign sequence analysis. Imagine, for instance, that an SR is in a corridor where the number of the first door sign is larger than the goal-door one, and this corridor has an increasing door sign sequence. Hence, in terms of the Growing Direction factor, the SR should not consider this path as promising, since it is not very likely that its door sequence contains the goal-door. Therefore, the proposed Growing Direction factor, semantic information inferred from the door sign sequence, is beneficial to our NSOS system, given that it indicates the door signs organization in a segment.

For each $m_i \in C$, the Growing Direction factor first calculates the angle in which the door sign sequence is increasing, $\theta_{inc}(S(m_i))$. To determine it, all the detected door signs of the segment $S(m_i)$ are considered, $L(S(m_i))$, as illustrated by Figure 4.3a. Then, for all possible pairs of two different door signs, one is larger than the other, the vector that connects them is computed. Figure 4.3b demonstrates an example for door sign number 1, and how the vectors are computed in pairs, such as (1,2), (1,3), (1,4), and (1,5). As it shows, the door sign number 1 has four vectors, while the door sign number 4 has only one. To illustrate, if we align all these detected door signs in the centre of the corridor, as shown by Figure 4.3c, it would be easier to understand that the sum of vectors from Figure 4.3b and the final $\theta_{inc}(S(m_i))$, indicate that the sequence increases to the right. To make this process even clearer, Figure 4.3d repeats the same procedure to the other door signs remaining, i.e., 2, 3, and 4. Here, it is important to mention that the door signs, i.e., the yellow circles, were represented within the white area to help the explanation. In practice, they appear within the grey area, as illustrated in Figure 4.2b.

Figure 4.4 illustrates a partial map from the simulator used in this chapter, and it helps to explain the importance of the Growing Direction. The robot has started at the intersection of three corridors, and it has chosen the corridor number 3, i.e., the one on the right. According to the robot's orientation in this corridor 3, the door sign sequence is considered increasing. Hence, in this current scenario, if the goal-door was 40, for instance, the Growing Direction factor would consider corridor 3 as promising. In contrast to this, the same corridor 3 would not be promising if the goal-door was 2, because the sequence only increases. It means that the distance from door sign 2 also increases as

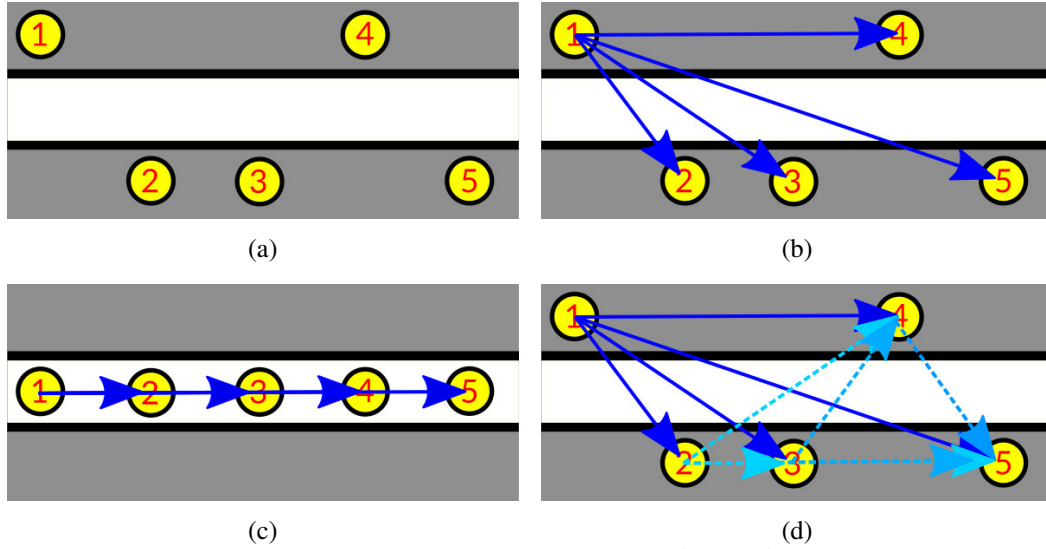


Figure 4.3 – Demonstration of how the increasing angle $\theta_{inc}(S(\mathbf{m}_i))$ is computed in a segment. All the detected door signs within the segment, (a), are considered to calculate the $\theta_{inc}(S(\mathbf{m}_i))$. The first step, (b), illustrates the vectors from door sign 1 to the other door signs, and it is easier to understand the effect of this vector calculation aligning all the door signs, (c). The final step of the vector computation, (d), shows all the vectors to all possible pairs.

the SR continues in that corridor and the SR will not get closer to the goal-door. Besides these two examples, which help to understand how the Growing Direction factor behaves, a third case is important to mention. Imagine that the goal-door was 21. This factor would be high until the SR recognizes the door sign 20 in corridor 3. However, after that its value would decrease as the SR goes on, and the door sign sequence increases. Hence, in terms of only the Growing Direction factor, the robot should continue its search in either corridor 1 or 2.

One possible solution to deal with the aforementioned third case is to measure whether all the door signs within the sequence are smaller or bigger than the goal-door. Hence, the amount of door signs that are smaller or bigger than the goal-door are counted by the functions $SL(S(\mathbf{m}_i))$ and $BL(S(\mathbf{m}_i))$, respectively. The factor $\zeta(\cdot)$ measures the possibility of a segment to have door signs smaller or bigger than the goal-door, defined by

$$\zeta(S(\mathbf{m}_i)) = \frac{(SL(S(\mathbf{m}_i)) - BL(S(\mathbf{m}_i)))}{\max(SL(S(\mathbf{m}_i)) + BL(S(\mathbf{m}_i)), w_g)}, \quad (4.9)$$

where $-1 \leq \zeta(S(\mathbf{m}_i)) \leq 1$. When $\zeta(S(\mathbf{m}_i)) = 1$, it means that in $S(\mathbf{m}_i)$ there are only bigger door signs than the goal-door. On the other hand, when $\zeta(S(\mathbf{m}_i)) = -1$, there are only smaller door signs than the goal-door. Lastly, when $\zeta(S(\mathbf{m}_i)) = 0$, both $SL(\cdot)$ and $BL(\cdot)$ are equal. w_g is a threshold used to control the minimum amount of detected door

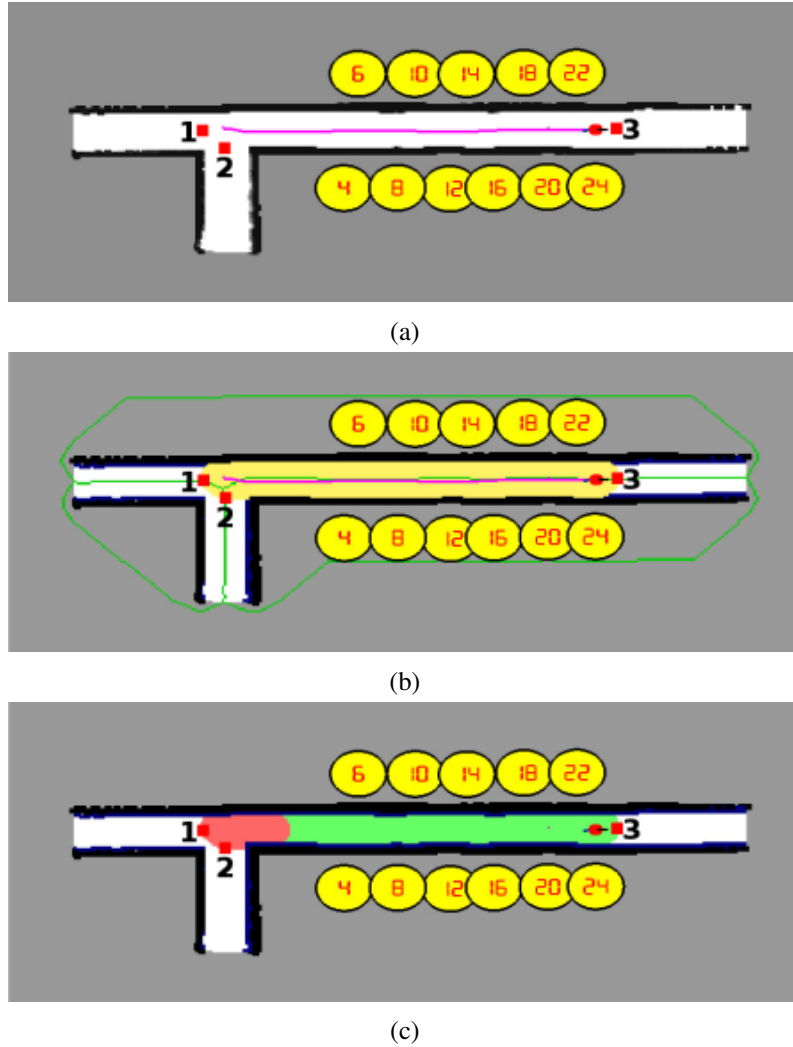


Figure 4.4 – Partial 2D map of the environment to show the importance of Growing Direction factor. All images represent the same part of the environment, but (a) shows the simple 2D grid map, (b) shows the visited region, and (c) shows the two segments of the map.

signs are necessary to this equation reaches 1 or -1.

In addition, Growing Direction factor also considers $\theta_f(\mathbf{m}_i)$. It is the Voronoi angle at cell \mathbf{m}_i . The difference angle between $\theta_f(\mathbf{m}_i)$ and $\theta_{inc}(S(\mathbf{m}_i))$, measured by $\gamma(\theta_f(\mathbf{m}_i))$, indicates whether $\theta_f(\mathbf{m}_i)$ is pointing to the same direction than $\theta_{inc}(S(\mathbf{m}_i))$. Then,

$$\gamma(\theta_f(\mathbf{m}_i)) = 1.0 + \left| \frac{\theta_f(\mathbf{m}_i) - \theta_{inc}(S(\mathbf{m}_i))}{\pi} \right| (-2.0), \quad (4.10)$$

where $-1 \leq \gamma(\theta_f(\mathbf{m}_i)) \leq 1$. When $\gamma(\theta_f(\mathbf{m}_i)) = 1$, it means that $\theta_f(\mathbf{m}_i)$ and $\theta_{inc}(S(\mathbf{m}_i))$ are pointing at the same direction. On the contrary, when $\gamma(\theta_f(\mathbf{m}_i)) = -1$, it means that they are pointing to opposite directions.

Now, the Growing Direction factor of a cell \mathbf{m}_i , $\varphi_g(\mathbf{m}_i)$, is defined as

$$\varphi_g(\mathbf{m}_i) = \frac{\zeta(S(\mathbf{m}_i))\gamma(\theta_f(\mathbf{m}_i)) + 1.0}{2.0}, \quad (4.11)$$

where $0 \leq \varphi_g(\mathbf{m}_i) \leq 1$. When $\varphi_g(\mathbf{m}_i) = -1$, it means that is less likely to reach the goal-door given how the door signs are set in $S(\mathbf{m}_i)$. Differently, when $\varphi_g(\mathbf{m}_i) = 1$, it means that is high likely. When $\varphi_g(\mathbf{m}_i) = 0$, it means that the Growing Direction factor is not sure about either the growing direction angle, or about the smaller and larger numbers. Hence, it cannot indicate whether \mathbf{m}_i is a very likely frontier.

4.3.3 Parity factor

This factor considers the characteristics of a door sign to be either *even* or *odd*, the kind of information that is not explicitly available in the environment. However, it can be easily inferred after the number recognition. The idea is to attribute a high probability to corridors that contain mostly door signs with the same parity as the goal-door. It is important to mention that this factor also considers the case in which a corridor contains both even and odd door signs since the probability is proportional to their respective amount.

To calculate the Parity factor, first it is necessary to count the amount of door signs from $S(\mathbf{m}_i)$ that have their parity alike or different than the goal-door. We use the functions $AL(S(\mathbf{m}_i))$ and $DL(S(\mathbf{m}_i))$ to count the alike and different parities, respectively. Then, for a cell $\mathbf{m}_i \in \mathbf{C}$, its Parity factor, $\varphi_p(\mathbf{m}_i)$, is given by

$$\varphi_p(\mathbf{m}_i) = 0.5 + \frac{AL(S(\mathbf{m}_i)) - DL(S(\mathbf{m}_i))}{\max(AL(S(\mathbf{m}_i)) + DL(S(\mathbf{m}_i)), w_p)} 0.5 \quad (4.12)$$

where $0 \leq \varphi_p(\mathbf{m}_i) \leq 1$, and w_p is a threshold used to control the minimum amount of detected door signs that are necessary to this equation reaches 0 or 1. When $\varphi_p(\mathbf{m}_i) = 1$, it means that all the observed numbers from doors sign in the segment where the SR have the same parity than the goal-door, whereas $\varphi_p(\mathbf{m}_i) = 0$ is the opposite. When $\varphi_p(\mathbf{m}_i) = 0.5$, it means that $AL(\cdot)$ and $DL(\cdot)$ are equal, and therefore is not possible to ensure the parity of the segment $S(\mathbf{m}_i)$.

4.3.4 Robot and Door Orientation factors

The SR moves through the environment, and it detects numbers from door signs as they are in its path. Usually, the position of doors in the environment follows a pattern that includes the possibility of existing doors only on horizontal or vertical corridors. Therefore, aiming to find the goal-door quickly, it is better to prioritize corridors in the same orientation than the already visited ones containing many doors. If the robot can prioritize the corridors in the same orientation, by consequence, its most common orientation will be an angle similar to these corridors.

To calculate the Door Orientation factor, it is considered a history of the λ_d most recent robot's orientations when a door sign was detected. Figure 4.5a illustrates an example, in which the robot's poses from **c** to **k** would be saved. Based on this history, it is computed a histogram of such orientations. Each bin saves the percentage of each possible robot's orientation. Then, given the orientation of \mathbf{m}_i , $\theta_f(\mathbf{m}_i)$, it is consulted in the robot's orientation histogram the probability of finding a door sign considering such orientation,

$$\varphi_o(\mathbf{m}_i) = H_d[\theta_f(\mathbf{m}_i)], \quad (4.13)$$

where $H_d[\cdot]$ is the door orientation histogram, and the Door Orientation factor is $0 \leq \varphi_o(\mathbf{m}_i) \leq 1$, in which 1 is 100% and 0 is 0%. The number of bins in the histogram $H_d[\cdot]$ defines its level of sampling. A $H_d[\cdot]$ with 360 bins provides a finer estimation than a $H_d[\cdot]$ with 45 bins. In our work, $H_d[\cdot]$ has maximum 179 bins, because we are not interested in saving the direction. Hence, if the robot's orientation is 0° or 180° , both will point to the same index in $H_d[\cdot]$ because they are in the horizontal line (although pointing to opposite directions).

The Door and Robot Orientation factors are very similar to each other. The difference between them is that the first one saves the robot's orientation only when a door sign has been recognized. Therefore, it prioritises the $\theta_f(\mathbf{m}_i)$ that has the highest $H_d[\theta_f(\mathbf{m}_i)]$, i.e. the orientation in which the robot has detected most of the door signs. On the other hand, the idea of the second one, the Robot Orientation factor, is to prioritize the $\theta_f(\mathbf{m}_i)$ that is most similar to the robot's orientation that is more frequent. It does not consider when the door signs were recognized. This factor makes the robot consider other paths that, despite not having door signs, may connect to other more promising ones.

As the robot moves through the environment, its λ_r most recent orientations are saved, as shown by the robot's pose, from **a** to **o**, in Figure 4.5a. They are used to compute

a histogram. Each histogram bin represents an angle and its percentage in the history of the robot's orientation. Given the calculated histogram, the $\theta_f(\mathbf{m}_i)$ is used as an index to get the probability of that angle, as presented by

$$\varphi_r(\mathbf{m}_i) = H_r[\theta_f(\mathbf{m}_i)], \quad (4.14)$$

where $H_r[\cdot]$ is the robot orientation histogram, and the Robot Orientation factor is $0 \leq \varphi_r(\mathbf{m}_i) \leq 1$. When $\varphi_r(\mathbf{m}_i) = 1$, it means that $\theta_f(\mathbf{m}_i)$ is an orientation that is equal to the unique robot's orientation saved. On the other hand, when $\varphi_r(\mathbf{m}_i) = 0$, it means that $\theta_f(\mathbf{m}_i)$ is an orientation that the robot did not do.

The scenario in Figure 4.5 illustrates the importance of both Robot and Door Orientation factors. The robot starts at the pose **a** in Figure 4.5a, and it moves forward until pose **p**. Then, it finds a second intersection between points 3 and 4, and it has to decide which one it should take. At this moment, the robot has detected door signs from pose **c** to **k**, with robot's orientation mostly at 0° as the reference shown in Figure 4.5a. Then, $H_d[0] = 100\%$. Besides, the robot has moved most of the time heading 0° , as illustrated by the 12 poses (from **a** to **l**) in Figure 4.5a, against the four poses (from **m** to **p**) heading approximately 270° . Then, $H_r[0] > H_r[270]$. Hence, when the semantic planner has to decide where to go, among the four available points, the 3 and 4 will have higher priorities due their proximity to the robot's pose. Besides, point 3 will have the highest priority among the two because of its orientation, which is 180° . In this case, $H_d[0] = 100\%$ (recall that 0° and 180° point to the same index in $H_d[\cdot]$) while point 4 is $H_d[90] = 0\%$ (converting 270° to the range $[0^\circ, 180^\circ]$ of $H_d[\cdot]$). Therefore, $H_d[\cdot]$ suggests that point 3 is the most promising one considering the organisation of the environment that has been observed by the robot.

4.3.5 Closeness factor

The fifth factor considered by our Semantic Planner is the distance between the robot cell \mathbf{m}_r and each $\mathbf{m}_i \in \mathbf{C}$, i.e. the smallest number of Voronoi cells that connects each pair of $(\mathbf{m}_r, \mathbf{m}_i)$. Its goal is to guide the robot towards the closest \mathbf{m}_i , instead of spending battery and time going to the farthest one. Take the Figure 4.6 as an example, and suppose that the goal-door was 71. The robot has started at the intersection between corridors 1 and 2, and it has moved to the right corridor, Figure 4.6a. After a few minutes,

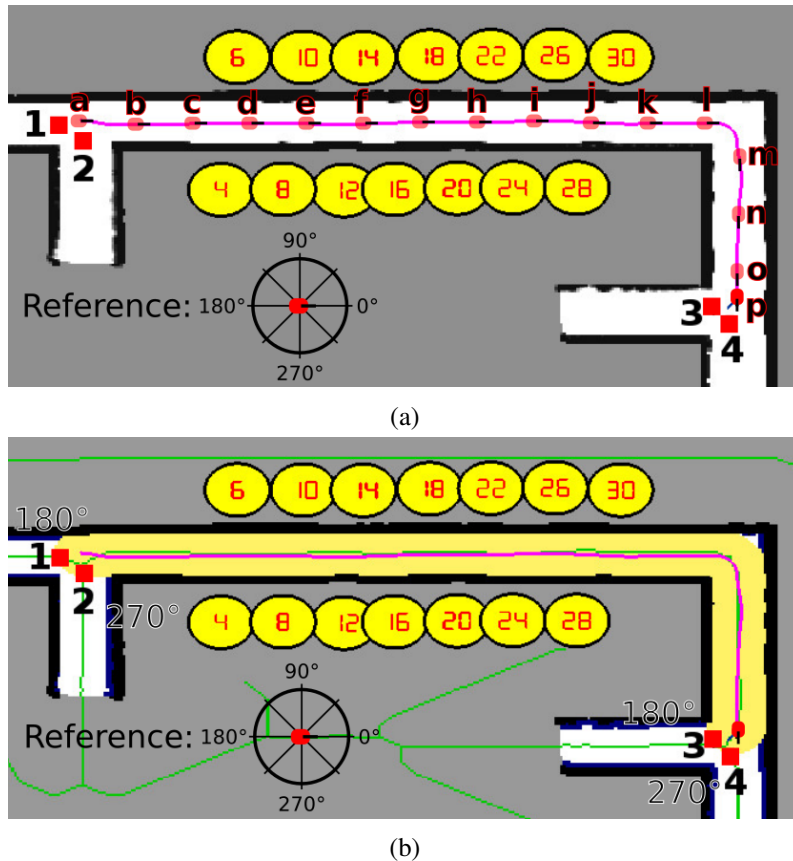


Figure 4.5 – Partial 2D map of the environment to show the importance of the Robot and Door Orientation factors. (a) shows the robot's orientation during its trajectory and (b) the orientation of each point.

guided by the Robot and Door Orientation factor, it has chosen to continue the searching on corridor 3. Even though this corridor has the same parity as the goal-door (both are odd), the Growing factor indicates that corridor 3 is not promising. Therefore, the robot should continue the searching in one of the other three options, corridors 1, 2, or 4. The Closeness factor is responsible for indicating the closest option to the robot, given by the sum of green cells that connect the robot's current position and each red point near the numbers 1, 2, and 4, as shown in Figure 4.6b.

The first step of the Closeness factor calculation is to find the smallest proximity m_j^{\ll} between m_r and all $m_i \in C$. It only considers the Voronoi cells in m , in which one cell is considered as one to the proximity sum. It is given by

$$m_j^{\ll} = \arg \min_{m_i \in C} (D(m_r, m_i)) \quad (4.15)$$

where $D(\cdot, \cdot)$ is the function that counts the number of cells between two other specific cells. In this factor, only Voronoi cells are counted, regardless they are within mapped or unknown regions.

$D(\mathbf{m}_r, \mathbf{m}_j^{\ll})$, and $\varphi_c(\mathbf{m}_i) = 0$ means that $D(\mathbf{m}_r, \mathbf{m}_i)$ is so high that makes the division be around zero.

4.4 Experiments and Results

This section presents the experiments carried out in simulation and in the real world. Section 4.4.1 explains the software setup used in our simulated experiments, as well as the differences between the physical and simulated experiments. Section 4.4.2 presents the results of the simulation phase, comparing the performance of our NSOS system and an entirely geometric OS system, called *Greedy*. Four different maps were considered in this comparison. Section 4.4.3 introduces a second type of comparison, in which these two initial OS systems, ours and the Greedy one, are compared to human participants teleoperating the robot in the simulation setup while performing the searching task. Finally, Section 4.4.4 demonstrates how our NSOS system performs in the physical world, as well as the information about where this test was performed.

It is also important to report the parameters used by our system throughout all the experiments presented below, either in simulation or in the real world. Both w_g and w_p are set to eight. This means that in Equations 4.9 and 4.12, respectively, the closer or higher to eight the number of detected door signs is, the more important the Growing Direction and Parity factors become. The number eight was chosen to balance the importance of the factors since a small number would make them important soon in the search process, and a large number would play the opposite role. In addition to these two parameters, the Robot and Door orientation factors also have some parameters. The size of the $H_d[\cdot]$ is four, which is the outcome of dividing the range of $[0^\circ, 179^\circ]$ by 45° . It means our approach considers the robot's orientations when detecting a door sign in groups of 45° (e.g., if the robot detects a door sign and its orientation is 42° , $H_d[0]$ is incremented). For the histogram $H_d[\cdot]$, we consider the past 6.000 orientations, as this sensor reading is noisy. For the case of $H_r[\cdot]$, we assume a finer setup since the robot may be in a different orientation in the range of $[0^\circ, 359^\circ]$. The size of $H_d[\cdot]$ is 18, and we consider the past 600.000 readings due to our high reading rate from the robot's orientation, the presence of noise in the data, and to reduce the impact of an unexpected turning that may happen.

4.4.1 Simulated Experiment Setup

The setup of the simulated experiments is represented in Figure 4.7. The MobileSim simulates a Pioneer 3-DX robot equipped with a 180° Laser, providing its odometry information and its laser sensor readings, Figure 4.7a. However, MobileSim does not provide information from door signs, which is vital for our experiments. Therefore, we developed a door simulator (DS) to mimic both the two RGB cameras that are embedded in the physical robot and the Image Processing module that recognizes the numbers, Figure 4.7c. DS provides the number of door signs and their positions in the world when they are within the robot's FoV. Then, the final setup combines the MobileSim to read the robot's information and our DS to provide the door signs information, as illustrated in Figure 4.7.

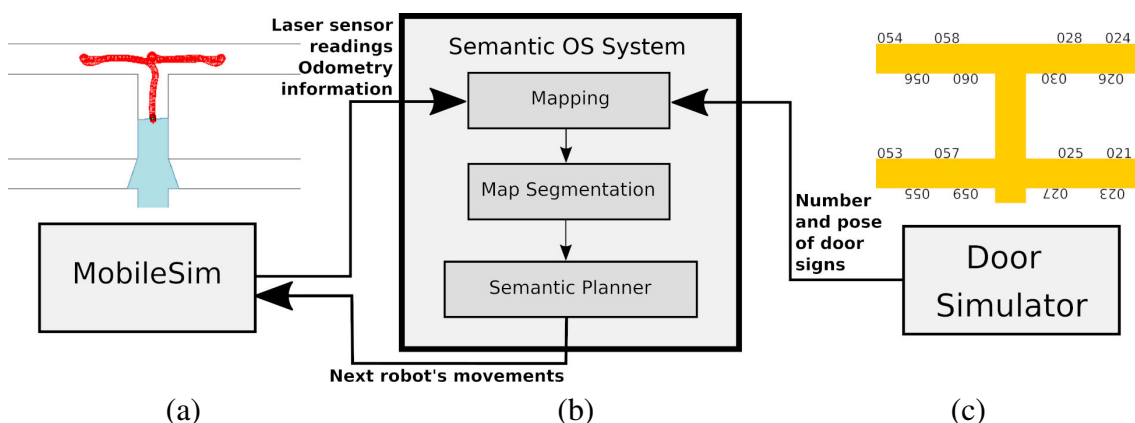


Figure 4.7 – Software setup used in the simulated experiments. It shows the MobileSim in (a), (b) represents the robot's and door signs information as input to our NSOS system that returns the robot's next movements, and (c) is the door signs map as ground-truth in the Door simulator.

Both (a) and (c) represent the same position on the map.

The first evaluation of our NSOS system was made through the comparison with the Greedy OS system in simulated indoor environments. The experiments considered four different scenarios. Table 4.1 and Figure 4.8 present the details of the scenarios. The four scenarios vary considerably regarding the number of door signs and how they are set, the size of the buildings, and the orientation of the corridor. The *Normal* and *Inverse* were made aiming to test the OS systems in scenarios with many long corridors intersecting each other, where the OS systems are forced to make decisions very often. Due to the high number of door signs in both scenarios, four of them were chosen as goal-doors for the tests, one in each horizontal corridor. Their difference is that *Normal*, Figure 4.8a, has its door signs sequence increasing from the middle to the borders. The *Inverse*, Figure 4.8b, is in the other way around. This way, we can test the performance

of our NSOS system in different door signs arrangements. Since the *Normal* and *Inverse* scenarios have been designed by us for this experiment and we would like to test the OS systems in real world environments, we also considered the *Hotel* and *KTH*. The *Hotel* is the third scenario used in the experiments, and it represents the third and fourth floors of the Hotel Pennsylvania (MCKIM, 1919) located in New York. It has the highest amount of door signs, along with a large environment containing many door signs and long corridors, Figure 4.8c. The *Hotel* scenario aims to test the OS systems in terms of how they perform when there is a large number of door signs in the environment. A bad choice in *Hotel* may cause a long run that will not lead to the goal-doors. The fourth scenario is from a public dataset called KTH Campus¹, Figure 4.8d, that contains more than 38,000 rooms in total, considering the many floor plans from different buildings (AYDEMIR; JENSFELT; FOLKESSON, 2012). Even though the particular floor plan chosen for this test, called *KTH* scenario, has the lowest amount of door signs compared to the three other scenarios used in the tests, it presents corridors in a different orientations, i.e., not only in horizontal or vertical. All tests in the simulation were carried out on a laptop with 8GB RAM and processor *i7*.

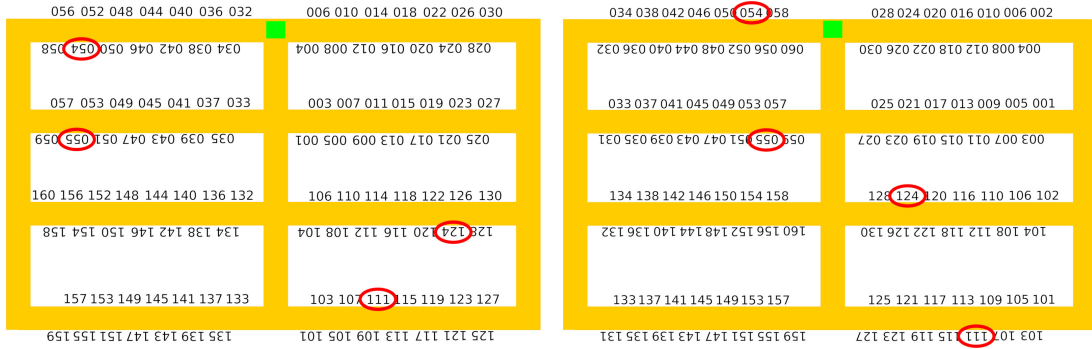
Table 4.1 – Different scenarios used on our simulated tests.

Name	# of Door signs	Goal-doors
<i>Normal</i>	113	54, 55, 111, 124
<i>Inverse</i>	116	54, 55, 111, 124
<i>Hotel</i>	124	76, 135, 148, 185
<i>KTH</i>	47	756

4.4.2 Semantic and Greedy Object Search systems

Our NSOS system was early introduced in Sections 4.2 and 4.3. In this section, the performance of our system is compared to the Greedy OS system, which has the exact same basis presented in Section 4.2, but its planner is composed only by the geometric factor from Equation 4.7, i.e. the Equation 4.7 with $\alpha = 0$. Therefore, both systems only differ by their planners, which are responsible for the reasoning over their inputs. In summary, the Greedy OS system searches for goal-doors based on the nearest frontier, whereas our NSOS system considers environmental information to make smarter deci-

¹It was used the left building from the floor plan identified as 0510028829_A30-00-07, A0043015. The dataset can be found at <http://www.csc.kth.se/~aydemir/KTH_CampusValhallavagen_Floorplan_Dataset.tar.bz2>

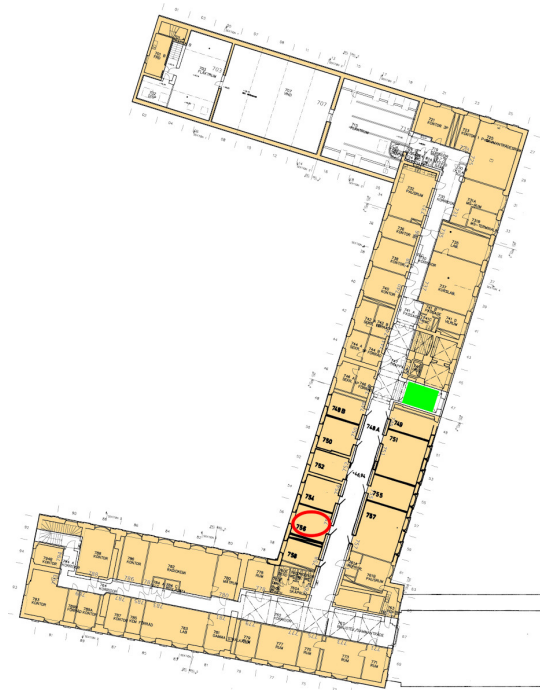


(a)

(b)



(c)



(d)

Figure 4.8 – The four maps used in the simulated experiments. The green squares represent the position where the robot has started, and the red circles highlight the goal-doors. The maps are (a) *Normal*, (b) *Inverse*, (c) *Hotel*, and (d) *KTH*.

sions.

Both systems were tested using the same simulation setup. For the four scenarios, the door signs shown in Table 4.1 were set as goal-doors. They were chosen to cover as many corridors of the scenarios as possible. For each goal-door, both systems have been run ten times each in these experiments, so we have statistically significant results. For every test, it was measured, in meters, the distance traveled by the robot from its initial position until it finds the goal-door. Since we consider the distance traveled as the search cost, the shorter the distance, the better is the system’s performance. Even though we have not measured and presented the search cost in terms of time, it is important to mention that both our NSOS system and the Greedy one moved the robot with the same

velocity, and at no moment the robot stood still, wasting time. Therefore, the system that provides the shortest distance traveled is also the fastest. Throughout the tests presented in this Section, in Equation 4.4 we used $\delta = 2$. This parameter mean that the map of the environment has been segmented into two tyoes, a corridor and not a corridor, as shown in Figure 4.1c.

Tables 4.2, 4.3, 4.4, and 4.5 present the results of both systems in each scenario in the simulated tests, *Normal*, *Inverse*, *Hotel*, and *KTH*, respectively. The colorful columns represent the results achieved by the tested OS systems. The first one is the result from the Greedy OS system, and the other three are from our semantic one. In the greedy column, the value 0,00% means that $\alpha = 0\%$ in Equation 4.7, and hence, the planner becomes fully geometric. In the semantic columns, the same value ranges from 80,0% until 100,0%, which means that the α ranges from 0.80 until 1.0 in Equation 4.7. Hence, it changes the importance of the semantic factor in that equation. Our NSOS system was also tested with α ranging from 0.5 up to 0.7, but the results were not significant and not presented in the tables. The rows of the tables correspond to the two OS systems' performance while searching for each goal-door. Each system's performance for every goal-door was evaluated in terms of Median, Average, Standard Deviation, Minimum, and Maximum distances. It is also important to highlight that within a row, the color of the table cells ranges from green to red. Green represents the cell with the smallest value within a row, and red represents the largest one.


The results in Table 4.2, *Normal* scenario, and in Table 4.3, *Inverse* scenario, are similar in terms of which column has the most red cells. In both cases, our NSOS system has a better performance than the greedy one, as most of the green cells are within the semantic columns, mainly when $\alpha = 80.0\%$ and $\alpha = 90.0\%$. In one of its ten runs for such map, the Greedy OS system made a sequence of decisions that lead it to find the goal-door 54 with the shortest distance (82.19 *m* in Table 4.2). This low result probably contributed to the lowest average being achieved by it, 121.94 *m* in the same table. However, it is also important to highlight that the standard deviation for the Greedy system for that goal-door is the highest one, 41.59 *m*. It means that it did not find the goal-door 54 traveling the shortest distance every test. A similar behaviour for this system can be seen when searching for goal-door 124, in which the minimum traveled distance was achieved by the Greedy system, 49.56 *m*, but not for all the ten runs (its standard deviation for this goal-door was the highest, 70.09 *m*). In Table 4.3, the lowest average and minimum of the goal-door 111 are from the greedy OS system, but again its standard deviation is the high-

est. It means that the ten tests of the greedy system vary considerably, as shown by the difference between its minimum and maximum values, 61.86 *m* and 278.57 *m*, respectively. On the other hand, the standard deviation within the semantic columns is lower, meaning our semantic system has a constant behavior during the searches. It guides the robot through a similar path through the ten runs, making the same decisions in different executions.

Table 4.2 – Results of the greedy and our NSOS systems in the *Normal* scenario. All the results are shown in meters.

Goal Doors	Value of α				Indices (m)
	Greedy	Semantic			
	0,00%	80,00%	90,00%	100,00%	
54	117,28	226,48	199,68	132,75	Median
	121,94	226,66	200,31	134,77	Average
	41,59	2,80	2,22	30,64	Std. Dev.
	82,19	221,42	197,88	100,40	Min
	230,01	231,11	204,13	186,80	Max
	132,09	72,44	64,20	114,96	Median
55	132,68	72,61	63,88	136,47	Average
	46,71	0,28	1,06	36,03	Std. Dev.
	74,45	72,34	60,89	113,42	Min
	243,17	73,18	64,44	225,75	Max
	171,78	236,78	137,85	60,86	Median
	170,69	243,37	135,02	67,34	Average
111	70,15	71,52	4,39	16,99	Std. Dev.
	65,99	136,81	128,22	46,73	Min
	275,52	383,46	138,75	108,38	Max
	196,31	61,88	59,75	67,86	Median
	148,50	61,86	63,31	67,05	Average
	70,09	0,10	11,24	17,08	Std. Dev.
124	49,56	61,68	59,64	49,58	Min
	200,36	62,01	95,30	86,16	Max

Color scale

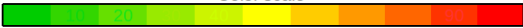


Smallest value of the row Biggest value of the row

Table 4.3 – Results of the greedy and our NSOS systems in the *Inverse* scenario. All the results are shown in meters.

Goal Doors	Value of α				Indices (m)
	Greedy	Semantic			
	0,00%	80,00%	90,00%	100,00%	
54	82,03	29,97	25,22	131,57	Median
	98,88	29,68	24,98	135,18	Average
	41,52	0,89	0,90	15,13	Std. Dev.
	72,61	27,16	22,44	118,57	Min
	207,20	30,03	25,40	166,40	Max
	152,67	69,26	41,35	128,48	Median
55	154,29	62,34	46,28	109,80	Average
	54,78	10,38	8,13	41,57	Std. Dev.
	63,60	49,44	41,11	35,00	Min
	232,54	71,31	61,34	138,67	Max
	273,64	250,07	219,97	214,02	Median
	200,16	230,27	205,45	209,96	Average
111	102,57	54,63	45,38	14,65	Std. Dev.
	61,86	153,58	124,08	188,42	Min
	278,57	292,28	267,79	231,98	Max
	205,26	177,16	160,49	173,34	Median
	165,73	145,59	137,05	156,20	Average
	82,65	42,73	38,69	60,25	Std. Dev.
124	58,15	92,99	80,53	35,59	Min
	291,36	181,82	162,30	200,52	Max

Color scale



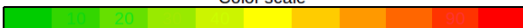
Smallest value of the row Biggest value of the row

The Tables 4.4 and 4.5, from *Hotel* and *KTH*, present similar results than the two previous tables in terms of the greedy column having most of the red cells. Besides, in general our NSOS system has better performance than the greedy system. Both tables also show that our proposed system is efficient in real scenarios. Even though the $\alpha = 100.00\%$ column within the semantic columns presents satisfying results, mainly in Table 4.5, a purely semantic OS system is not always suitable for searching tasks. The geometric factor in Equation 4.7 is essential and, combined with the semantic factor, may provide the best results. The Greedy system presents good results for all the four goal-doors in terms of minimum traveled distance. Again, as this system does not repeat the sequence of decisions during the search over all the ten runs, sometimes it is lucky enough to make a sequence of decisions that lead it to find the goal-door very quickly. The lack of consistency and robustness on Greedy's performance is demonstrated by its poor results in terms of standard deviation, being the worst of all for all goal-doors in Tables 4.4 and 4.5.

Table 4.4 – Results of the greedy and our NSOS systems in the *Hotel* scenario. All the results are shown in meters.

Goal Doors	Value of α				Indices (m)
	Greedy	Semantic			
		0,00%	80,00%	90,00%	
76	341,49	190,95	168,68	114,34	Median
	314,90	184,36	172,17	125,41	Average
	131,74	21,80	11,11	25,96	Std. Dev.
	143,97	123,07	167,93	96,47	Min
	491,69	199,53	203,74	171,91	Max
135	552,44	285,57	283,22	318,24	Median
	433,75	282,87	304,28	326,52	Average
	194,93	9,18	39,92	27,01	Std. Dev.
	93,17	256,95	280,69	292,30	Min
	555,11	287,15	380,32	374,25	Max
148	554,20	151,65	368,56	405,21	Median
	525,30	151,64	369,44	395,57	Average
	167,12	0,75	1,72	30,34	Std. Dev.
	114,65	149,97	368,09	327,40	Min
	671,81	153,01	372,60	434,30	Max
185	132,18	130,84	130,94	92,01	Median
	167,09	130,87	131,00	102,94	Average
	166,77	0,22	0,29	57,33	Std. Dev.
	51,79	130,57	130,71	51,72	Min
	552,68	131,22	131,55	193,82	Max

Color scale



Smallest value of the row Biggest value of the row

It is worth to mention that the shape of the environment and the door signs arrangement play an important role in our NSOS system's performance. Depending on the direction that the sequence of door signs number are increasing, for example, it is possible that our system performs differently. Both *Normal* and *Inverse* scenarios illustrate this situation, where the only difference between them is the direction in which the door signs number increase. Analysing the three semantic columns in both Figures 4.2 and 4.3, it is

compared to their optimal solution. For all goal-doors and systems tested in the simulation experiments, Table 4.10 shows how many times, in percentage, the averages are larger compared to the optimal solutions. For the case of our NSOS system from Tables 4.6, 4.7, and 4.8, it is considered the lowest average between the ones from $\alpha = 80.0\%$, $\alpha = 90.0\%$, and $\alpha = 100.0\%$.

Table 4.7 – The average and the standard deviations from the *Inverse* scenario, Table 4.3, and the optimal solution (shortest path) between each goal-door and the starting position. All the results are in meters.

Goal Doors	Value of α				Shortest Path (m)
	Greedy	Semantic			
	0,00%	80,00%	90,00%	100,00%	
54	98.88 ± 41.52	29.68 ± 0.89	24.98 ± 0.9	135.18 ± 15.13	7,65
55	154.29 ± 54.78	62.34 ± 10.38	46.28 ± 8.13	109.8 ± 41.57	15,98
111	200.16 ± 102.57	230.27 ± 54.63	205.45 ± 45.38	209.96 ± 14.65	40,67
124	165.73 ± 82.65	145.59 ± 42.73	137.05 ± 38.69	156.2 ± 60.25	24,65

Table 4.8 – The average and the standard deviations from the *Hotel* scenario, Table 4.4, and the optimal solution (shortest path) between each goal-door and the starting position. All the results are in meters.

Goal Doors	Value of α				Shortest Path (m)
	Greedy	Semantic			
	0,00%	80,00%	90,00%	100,00%	
76	314.90 ± 131.74	184.36 ± 21.80	172.17 ± 11.11	125.41 ± 25.96	63,16
135	433.75 ± 194.93	282.87 ± 9.18	304.28 ± 39.92	326.52 ± 27.01	72,51
148	525.30 ± 167.12	151.64 ± 0.75	369.44 ± 1.72	395.57 ± 30.34	75,81
185	167.09 ± 166.77	130.87 ± 0.22	131.00 ± 0.29	102.94 ± 57.33	52,53

Table 4.9 – The average and the standard deviations from the *KTH* scenario, Table 4.5, and the optimal solution (shortest path) between each goal-door and the starting position. All the results are in meters.

Goal Doors	Value of α				Shortest Path (m)
	Greedy	Semantic			
	0,00%	80,00%	90,00%	100,00%	
756	155.03 ± 25.82	33.41 ± 10.35	28.88 ± 4.6	24.52 ± 1.98	18,35

To compute the percentages presented in Table 4.10, it is considered the optimal solution of each goal-door for each scenario as 100%. Hence, if the average is larger than the shortest path, it will be higher than 100%, as in the case of the goal-door 54, scenario *Normal*. The greedy system is approximately seven times larger than the optimal solution, i.e., 709.39%.

In Table 4.10, most of the lowest percentages are within the semantic rows. There are few goal-doors in which the greedy system presents the lowest rate. That is the case of goal-door 54 of the *Normal* scenario, and the 111 of the *Inverse*. However, even though the greedy system presents low values, the values from the semantic system to the same goal-doors are similar. On the contrary, analyzing the goal-door 54 of the *Inverse* scenario, for instance, the value from the greedy system is almost four times larger.

The two OS systems were submitted to search for one goal-door at time in all scenarios. Hence, we do not have their results as if they were searching for a sequence

Table 4.10 – Comparison of the optimal solution (shortest path) of each goal-door from each scenario, with the averages from Tables 4.2, 4.3, and 4.4. The shortest path is equivalent to 100%, and the figure shows how large the averages are in comparison with the optimal solution.

Scenario	Exploration Approach	Goal Doors			
		54	55	111	124
Normal	Greedy	709,37%	497,45%	456,02%	447,83%
	Semantic	783,94%	239,48%	179,91%	186,55%
Inverse	Greedy	1292,42%	965,46%	492,16%	672,33%
	Semantic	326,54%	289,61%	505,14%	555,98%
		76	135	148	185
Hotel	Greedy	498,58%	598,19%	692,92%	318,12%
	Semantic	198,56%	390,11%	200,03%	195,98%

of goal-doors in a single attempt. However, we can speculate their performance in such conditions, based on their results in the previous experiments. In general, our NSOS system finds the first goal quickly, demonstrating that it is efficient even when there is no information available *a priori* about the environment. After finding the first goal-door, NSOS will be partially aware about the organisation of the environment, which may give it advantages for more accurate searches in the future. On the other hand, in most of the cases the Greedy system wastes too much robot's resources to find the first goal-door of the sequence. In this process, it may discover the next goal-doors of the sequence and improve its performance a little bit. However, if the next goal-doors were not found during the first search, its performance will be even worse than our NSOS system in such a scenario.

4.4.3 Human participants performance in object searching task

The results presented in Table 4.10 illustrate how many times, in percentage, the results of the greedy and our NSOS systems are larger than the optimal solution. Some results from the semantic system are two, three, or even four times larger, whereas the greedy system provides results that are up to 12 times larger than the optimal solution for the scenario *Inverse* and goal-door 54.

Given only these high percentages, it seems that both systems are not suitable for the task of finding a target door sign in an unknown environment based on text information as visual cues. However, it is important to highlight that this task is challenging because the environment is unknown, and there is no way of planning an optimal path a priori. This section illustrates the difficulty level of the searching task by presenting an experiment in which human participants were invited to perform the searching while piloting the robot in the simulator presented in Section 4.4.1. This experiment measured (in meters) the

distance traveled by the robot from the initial position until the goal-doors. The distance traveled is the search cost used in this work, and hence, the shorter the distance, the better is the system performance.

Instead of using the planner of either our semantic or the greedy OS system to find the goal-door task, ten human participants were invited to teleoperate the robot in the simulation setup to perform the same role as our semantic planner. The participants were presented to the searching task beforehand, with a time to get familiar with the robot control system and our simulation setup. This experiment aimed to measure human performance in the same setup as the other tested system to show whether human reasoning provides better results than our NSOS system in the same conditions. Therefore, the humans controlled the robot in the same simulator and graphical interface as the two OS systems, as shown by Figure 4.1a. The difference of this experiment to the one from Section 4.4.2 is how the choices are made. In this case, the participants must choose where the robot must go, playing the planner role to choose the path.

In this experiment, only two scenarios were tested, with one goal-door each. The *Normal* and *Inverse* scenarios are certainly similar, differing only on how the door signs are arranged. Given that humans can memorize what they have seen, it would be unfair to submit them to two similar scenarios or more than one goal-door for the same scenario. Therefore, the *Normal* and *Hotel* scenarios were used for this experiment. Door signs picked as the target were 111 to *Normal* and 148 to *Hotel*. They are not too far nor too near to the robot's initial position. Hence, the participants would have to explore at least a small part of both scenarios. Throughout all the tests, the only data considered for the evaluation was the traveled distance.

Table 4.11 summarises the analysis of the ten participants. Besides, it also compares human performance to the greedy and our NSOS systems. As in the previous tables, green represents the cells with the smallest value within a row, and red represents the largest. As can be seen, our system presents a smaller average in both goal-doors, with the lowest indices compared to the others. The minimum traveled distance for the goal-door 148 is the only case where our NSOS system does not have the lowest result. For that result, one of the ten participants made some decisions that luckily brought them to the goal-door with the shortest traveled distance. The second shortest traveled distance from the participants was 199.23 *m*, a result two times larger than the 75.23 *m* shown in Table 4.11, and which supports that humans do not have a regular and efficient performance.

Table 4.11 – Human performance to the problem of OS system. It is compared to the greedy and semantic systems, in which all of them had to find two goal-doors, one in each scenario. The results are presented in meters.

Scenario	Goal Doors	Different approaches			Indices (m)
		Humans	Greedy	Semantic	
Normal	111	109,06	171,78	60,86	Median
		103,38	170,69	67,34	Average
		34,00	70,15	16,99	Std. Dev.
		63,88	65,99	46,73	Min
		170,94	275,52	108,38	Max
		261,17	554,20	151,65	Median
Hotel	148	299,30	525,30	151,64	Average
		136,10	167,12	0,75	Std. Dev.
		75,23	114,65	149,97	Min
		531,96	671,81	153,01	Max

Color scale

Smallest value of the row Biggest value of the row

In contrast to our NSOS system, the greedy one presents the worst results for both goal-doors, which confirms the previous results presented in Section 4.4.2. It is also worth mentioning the high standard deviation of the Human participants for both goal-doors. It shows that they had different performances, in which some had a shorter traveled distance than others. Some participants did not follow the same pattern when making decisions throughout their run. At the end of their participation, they have reported that they did not have an efficient strategy to search for the target, and no reasoning was made based on the door signs. One specific participant forgot, for only a few seconds, the goal-door for the *Hotel* scenario. The participant mentioned that as soon as they realized they could not remember the number, they look at a paper that was in front of them to read it again. The human eye has a wider FoV than the two cameras used in this experiment setup, which provides advantages to humans when searching for objects in an unknown environment. However, as previously mentioned, the goal of this experiment was to test humans as the decision-makers within the system. That is why they used the same software setup for the experiments as our semantic and greedy systems.

Besides the lowest average from the NSOS system, there are other advantages compared to the participants' results. Its slight standard deviation means that the same decisions were taken in all the ten test repetitions, which suggests that our system does not make random decisions. On the other hand, the same does not apply to the Human results, which means that every participant had their particular reasoning to make a decision. Hence, some participants are more efficient than others in this OS task. Besides the standard deviation, another advantage is that SRs are not disturbed by other moving objects or agents in the environment. Therefore, they can focus on the task, and they do not forget the goal-door, what happened to one of the humans during the experiment.

4.4.4 Experiments Using a Physical Robot

The experiment with a physical robot was performed in one of the buildings of the Federal University of Rio Grande do Sul, Brazil, where the Phi Robotics Research Lab is located. Figure 4.9 shows how this building is organised, as well as the rooms and their door signs. The robot used in this experiment is a Pioneer 3DX from MobileRobots, which is equipped with a LiDAR laser scan of 180° and two RGB cameras, as shown in Figure 4.10.

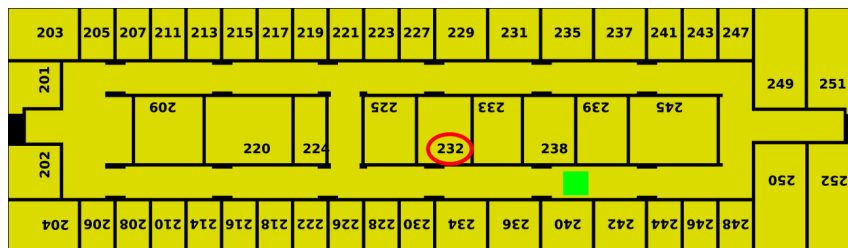


Figure 4.9 – Map used in the experiments with the real robot. It is the building where the Phi Robotics Research Lab is located at the Federal University of Rio Grande do Sul, Brazil. The green square represents the position where the robot starts, and the red circle highlight the goal-door.

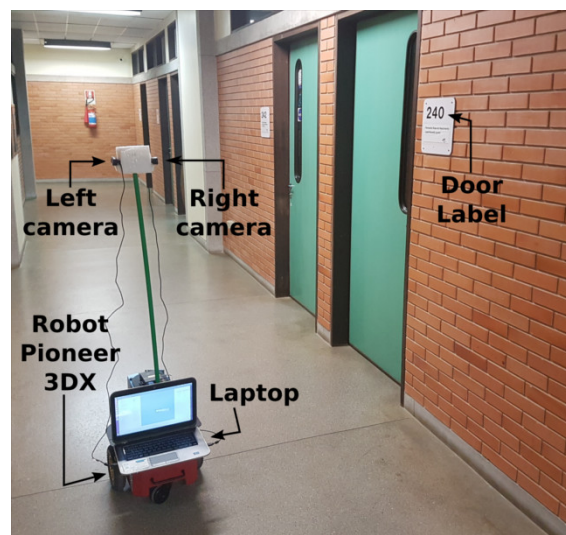


Figure 4.10 – The Pioneer 3DX robot used in the real environment experiment, as well as the two embedded cameras. The door signs of the environment are also depicted on this figure.

This experiment aims to demonstrate that our NSOS system works in physical scenarios, meaning that the robot should be able to find the target goal-door traveling the shortest distance as possible. For this experiment, goal-door 232 is chosen as the target, which is located on the left side of the initial position, the green square in Figure 4.9. Even though it was at the same corridor as the initial position, the experiment setup is suitable to prove that our system can make estimations over the detected door signs. From the

initial position, the SR can turn to the left or the right. If it decides the left direction, it will find the goal-door quicker, but the other option would take it to the opposite side. For this case, as soon as a few door signs have been detected, our semantic system would be able to reason over them and infer that this direction is not promising. Hence, it should change to the opposite direction. Therefore, this would show that our semantic system uses the door signs to find the goal-door efficiently.

When submitted to finding the goal-door 232 in a physical environment, the performance of our NSOS system was similar to the situation described above. Figure 4.11 depicts six steps of the system, and all of them show important moments for the searching. In Figure 4.11a, the robot just had finished one complete rotation to map its surroundings, and then it detected the door sign 240. In Figure 4.11b, the robot had chosen to turn right, where the door sign 242 has been found. This figure demonstrates how our planner decides on changing: *i*) the door signs 240 and 242 were recognized in an increasing sequence, and it means that as the robot goes forward, the distance from the goal-door increases; *ii*) the robot is between two frontiers, and even though the robot is closer to the one that is in front of it, the other is not that far from it; *iii*) the first two door signs were found in a horizontal corridor, and so far, the horizontal corridors are the more likely ones to contain other door signs. Combining all this information, the decision is that the frontier that the robot is following is less promising than the other one that is behind it. In Figure 4.11c, the robot has changed its orientation to the opposite one. The door sign 238 was recognized, which supports the orientation change. In Figure 4.11d, as the robot has not recognized any other door sign that contradicts its decision, the searching continues towards the left direction. In Figure 4.11e, it recognizes the door sign 234, which indicates a decreasing sequence towards the goal-door. Finally, in Figure 4.11f, the robot detects the goal-door 232, and the robot finishes the searching.

4.5 Summary

We proposed NSOS, a semantic OS system that relies on organisational semantic information inferred from numbers within the environment. The proposed system aims to demonstrate that it is possible to take advantage of numbers from textual signs to estimate the organisation of the environment and improve the robot's performance. Even though we have not tested our system with traffic signs or outdoor advertisements, for example, and only with door signs, the results show that detecting complementary information

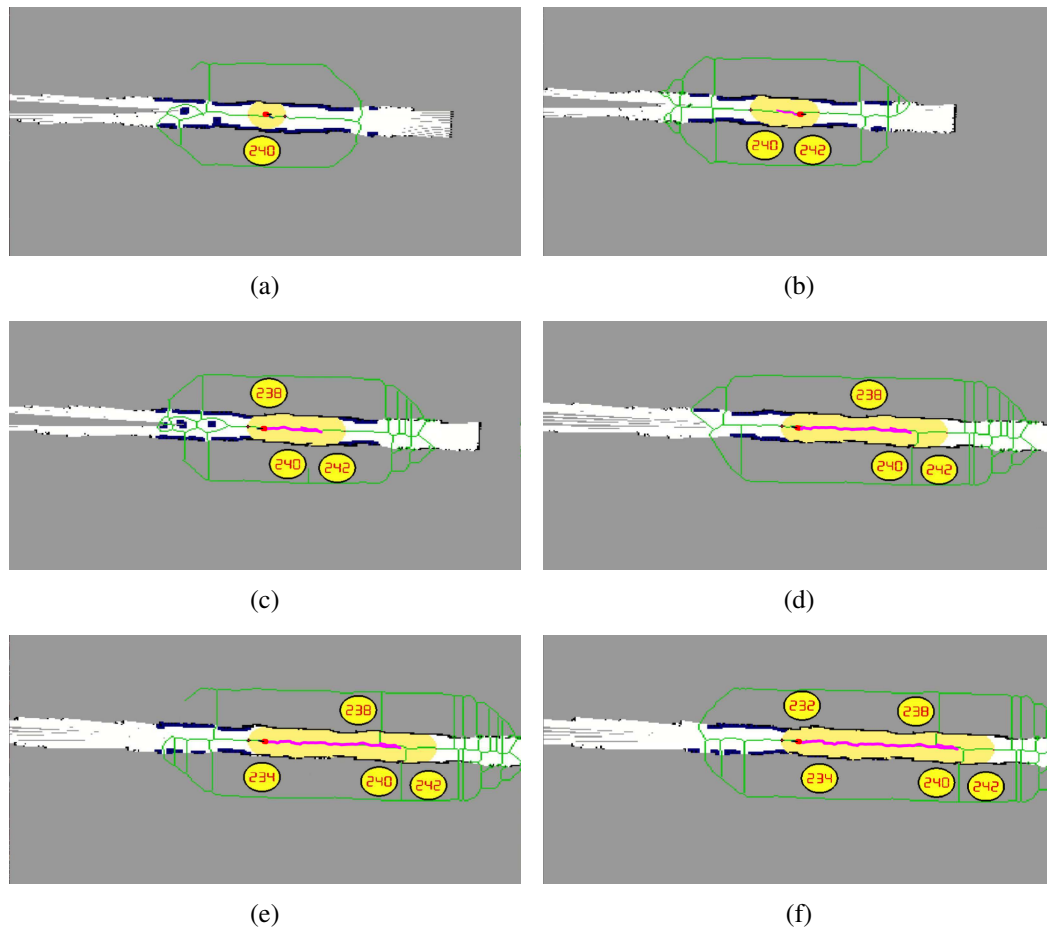


Figure 4.11 – Step-by-step of the performance of our NSOS system in the physical environment. *a* shows the initial position, where the robot has started the searching, whereas *f* shows the final step when the robot has found the goal-door 232.

from signs is promising for robotic solutions. Besides, our NSOS system also intends to encourage the mobile robotics research community to explore the advantages of semantic information for mobile robot tasks. To the best of our knowledge, there is no other OS system in the literature that extract semantic information from any of the aforementioned examples. Hence, we could not present a fair comparison between our proposal and other published method, other than the comparison we did with the Greedy OS system.

The main contributions of this chapter are:

- a robust semantic planner, based on five different factors, that reason over the door signs to find the goal-door with the SR traveling the shortest possible distance;
- a semantic OS system called NSOS which, by using our semantic planner, can make estimations over the detected numbers from door signs and estimate the organisation of the environment, and make an SR avoid continuing searching on non-promising regions;
- an analysis of the usage of information inferred from numbers as input to the seman-

tic planner within our NSOS. In general, the analysis shows that NSOS presented better results than human participants, both in the same simulation setup.

Our semantic planner applied to the OS system presented an excellent performance, mainly when compared to the results from the greedy system and humans performing the searching. Besides providing the shortest distance traveled, and by consequence, it was also the fastest search system, given that the SR moved with the same velocity in all the experiments. It is important to mention that no experiments were conducted in an environment where its rooms were randomly signed. This is because we believe that in such kind of environment, probably not even humans would be able to rely on the random signs and efficiently search for the target door sign, and our NSOS system would rely on an input that is not reliable. On the other hand, despite not being random, the *Hotel* map presents very challenging door sign configurations. Some corridors have a door sign sequence that does not increase or decrease, which does not have a predominant parity. These conditions do not reflect a well-structured environment, but our proposal still presented a robust performance, with better results than the other tested approach and humans. Our NSOS system does not support door signs labeled with other characters other than numbers. However, this limitation could be overcome with the development of different heuristics (or factors) that model the organisation of other labeling patterns. The modular characteristic of our semantic planner from NSOS allows the replacement of factors by new ones designed for interpreting specific labels. All the other parts from NSOS could remain the same.

Our semantic planner does not require that the door signs of the environment are arranged according to a specific pattern, as confirmed by the wide variety of the four tested scenarios. The scenario *Hotel* demonstrates how different the door signs can be located, and according to the feedback from the participants after their participation, the *Hotel* is indeed a little bit confusing for them.

The results show that our NSOS system presented better or similar results than the ones from human participants. However, it is important to highlight that these results were obtained when humans piloted the robot in the same simulation setup as the other experiments. The human eyes have a wider field of view than cameras, so in this case, it would be unfair to compare the performance of humans against robots if they had different visual sensors. That is why only human reasoning was considered in the experiments of this chapter.

5 LSOS: USING CHANGES IN THE ORGANISATION OF THE ENVIRONMENT OVER TIME

We have already seen the advantages of using organisational semantic information in OS tasks. The previous semantic information is inferred from both the organisation of door signs and the corridors of a building, which rarely change over time. In most of the time, rooms are identified by signs that last for many years, as well as the corridors that only change when the environment goes through a renovation. Our previous OS system, NSOS (MANTELLI et al., 2021), Chapter 4, estimates the organisation of indoor environments for OS tasks based on static source of data to infer the organisation semantic information. However, it is no surprise that SRs sometimes operate in semi- and fully-dynamic environments, where some objects and obstacles move occasionally, e.g. chairs in the launch area, or very often, e.g. the glasses of an older adult in their house. Supposing that an SR has to interact with some objects within the room that it has found with aid our our NSOS system from the previous chapter, it would be interesting that the SR could efficiently search for these objects. In this chapter we discuss how an OS system could estimate the organisation of the environment even if it is not static as in Chapter 4.

5.1 Proposal and contributions

In this chapter, we propose a Long-term Semantic OS system (LSOS) that estimates which regions of the dynamic environment are more promising to find a target object. The search is based on the organisation of objects throughout a period of time. We assume that human activities have an influence in the objects' placement (ZHOU et al., 2020). Besides, we argue that a robust OS strategy should consider the semantic information of how humans interact with objects over time, rather than searching objects only based on their shape or color. Instead of filtering the changes within the environment, the OS system could model and incorporate the changes in the objects' position over time to make predictions about its future positions (KRAJNÍK et al., 2020). For example, a person may move their smartphone many times throughout a day, but there are high chances that it will be on the bedside table during the night. Suppose the SR is tasked with finding the smartphone at night, and the OS system can understand this pattern. In that case, it will reduce the regions to search for only the bedroom. Consequently, it would improve

the robot's overall performance during the search.

Our LSOS system works in two modes. One is called recording, and it collects objects' data in the unknown environment, whereas the other is called requesting, and it is responsible for processing the saved data to perform the search. In the recording mode, the SR gathers information about the surrounding objects over a period of time, and then it builds the map of the environment (or just update it in case it already exists). The mapping could be done while the SR carries out other tasks, such as cleaning the floor or watching an older person. The requesting mode is executed when someone requests the SR to find a target object, in which our LSOS processes the gathered data to estimate where the object might be at the request moment. We use a Heat map to represent the probability of finding the target object, in which areas where are more likely to have the target object located are more heated than elsewhere. Before we continue, it is important to highlight that in the previous chapter our NSOS actually searched for objects and not just numbers from door signs. The numbers are the identifying code for the doors, which were the real object that our NSOS system meant to find and that the numbers were associated with. Therefore, even though the numbers of the door signs may not be a real object, in the end our previous NSOS system was searching for an object, as our next LSOS system does. The difference is that NSOS searched for only one type of object, doors, whereas LSOS can search for different sorts of objects, as we will see in this chapter.

Figure 5.1 illustrates how the core of our LSOS system, the requesting mode, works. For this example, imagine that an SR has already moved through the whole area performing the recording mode. Then, it built the 2D grid map and recorded the location of objects and the hour they have been detected. Saving the detection hour of an object, along with its position, is the core of our LSOS system. By doing this, our system will be able to estimate the organisation of the objects over time in the environment, and efficiently find the target object. It is as if the system could learn the objects arrangement in different moments of the days, and recognise patterns. Back to the example, when the SR receives a request to find a mug at 14:00, as shown in Figure 5.1, the requesting mode of LSOS uses the recorded information that a mug has been detected twice. The first time, at 9:00, it was in region A, and it was moved to region B at 13:00. The recording mode compares the current time, 14:00, with the two detection hours of the object. Then, it decides which region it should start searching. As region B has the smallest difference to the querying time, i.e. the most recent time the object has been seen it was in region B, the SR goes to that region first. The idea is that the more data our LSOS records, the more

precisa are its estimations.

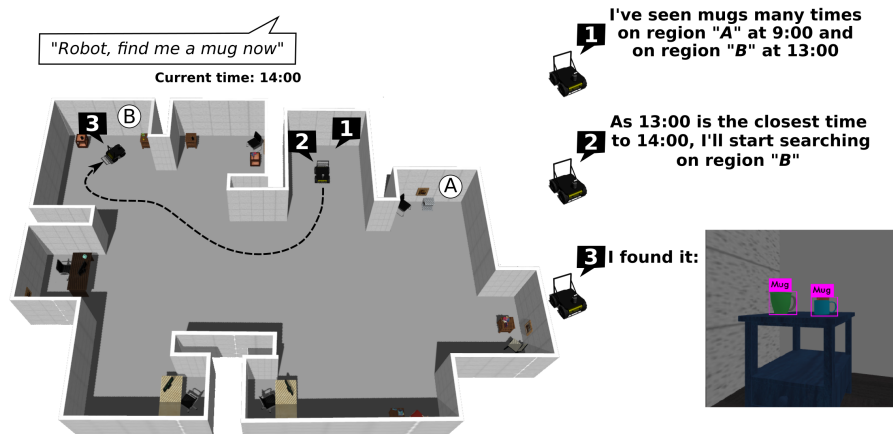


Figure 5.1 – One example of our proposed LSOS system operating in our custom made simulated environment. The robot's task is finding a mug at 14:00.

The contributions of this chapter are presented below:

- **Semantic OS system combined with a heat map to indicate the object's presence:** a probabilistic OS system that does not require a priori map from the environment, a training phase, nor data annotation. It builds a 2D grid map of the environment to estimate the robot's pose and indicate the free cells regions for navigating. Simultaneously, it builds a Heat map based on the 2D grid map, representing the computed estimations of the objects' likelihood for each area. For our LSOS system, a 3D map would not be favorable, as its benefits do not make up for the computational cost of building and updating it. A simple 2D is already suitable for our LSOS needs. However, depending on the secondary tasks the SR has to perform simultaneously to the OS task, such a 3D may be mandatory. In such case, our LSOS system will continue working without considerable adaptations from the 2D to a 3D map.
- **Self-contained and easy to deploy OS system:** a semantic OS system that works independently. It does not depend on a specific SLAM system for mapping the environment or an object detection module for detecting the objects. It only requires the robot's and objects' pose within the map, the objects' class, and the hour in which detection happened, which will be stored during the recording information mode.
- **Changes in the objects arrangement over time as an OS strategy:** a new strategy for OS that takes advantage of the organisational semantic information inferred from the objects arrangement throughout a period of time. It aims to reduce the

amount of regions of the environment that the SR has to search in, and improve the overall results. It saves information about *when* the objects were *where*, and then estimates the map regions' likelihood of containing the target object.

- **The organisational semantic information in dynamic environments:** in contrast to most of the works in robotics, which assume the world is static and filter the dynamic agents, our LSOS system incorporates the changes in the objects' position in the environment to improve the robot's performance in OS tasks. We present an analysis of the advantages of using such type of information as input to our system, which permits an efficient estimation of the region which is more likely to contain the target object.

5.2 LSOS system's search strategy

Our LSOS system is based on spatio-temporal information, and it requires data from the past to perform OS tasks efficiently. It is important to highlight that although our LSOS system depends on previous observations of the objects to compute the estimations, no data must be provided to LSOS *a priori*. Nevertheless, that is the only requirement of our system. The more observations the recording mode saves, the more precise are the estimations from the requesting mode. Our LSOS system does not depend on ambient sensors like in (SPRUTE et al., 2017) or environmental changes. Even though smart houses may have visual sensors that could be used to expand the SR's FoV, an OS system that depends on such set of sensors restricts its usage in other regular houses. Therefore, we focused on making our LSOS system self-sufficient, resulting in a two-mode recording and requesting system. Besides, our system can run simultaneously with any other task the SR performs, such as cleaning the environment or monitoring an old adult. Hence, a single SR can perform multiple tasks along with our LSOS system.

The first mode, recording, is responsible for gathering and saving all the data that our LSOS system needs, as Figure 5.2 illustrates. The goal of this mode is to fulfill our OS system's requirements, as mentioned earlier. In summary, the recording mode relies on an object-detection algorithm and a SLAM system. We use YOLO for object-detection (see Section 2.2.2), as it is a robust and popular package for such task, and apply the Gmapping package, a laser-based SLAM for building a 2D grid map, as step 1 in Figure 5.2 (BJELONIC, 2016–2018; GERKEY, 2013–2020). During the execution of this mode, when an object is within the camera's point of view, YOLO detects it, as

step 2 in Figure 5.2. A depth camera with a point cloud measures the distance between the robot and the object, and then our system calculates its position in the map given the robot's pose, as step 3 in Figure 5.2. Once the robot's and object's position are known, our system saves this information, along with the detection hour and the class of the object provided by YOLO, as step 4 in Figure 5.2. This recording mode can be executed hourly, daily, or while the robot performs any other task. It is also important to highlight that our OS system is not restricted to YOLO and Gmapping. Any object-detection and SLAM package, respectively, provide the same data type our system saves, and hence, could be used in our system.

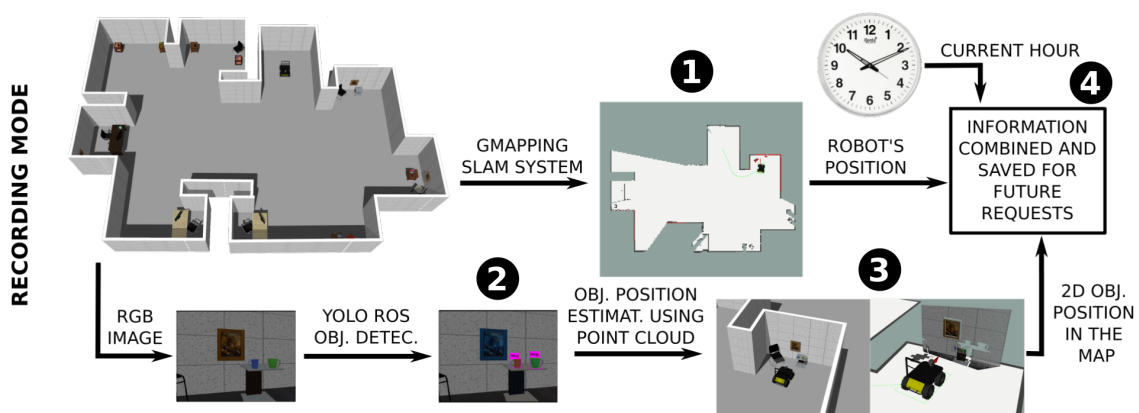


Figure 5.2 – Flowchart that explains how the recording mode from our LSOS system works. The mode gathers information about the environment and save it.

The second mode, requesting, is performed by our LSOS system when the SR is active with finding an object, i.e., when someone requests it to look for a target object. It worth to mention that when when our system is requested to search for a target object, it does not differentiate the instances of that object. For example, when searching for a book, both *The Lord of the Rings* and *Don Quixote* (or any other book) are valid options to LSOS. Hence, the target object represents a class of objects, and finding any instance that belongs to that class is sufficient. Furthermore, this mode aims to find the target with the robot traveling the shortest distance to save time and the robot's resources. All the data gathered in the recording mode plays an important role here, as it helps to improve the estimations of our LSOS system. Figure 5.3 illustrates the entire requesting process. The requesting mode builds a heat map out of the recorded data to estimate which environment regions are more promising to locate the target object in, as steps 1 and 2 in Figure 5.3. In contrast to many OS systems in the literature, our proposal considers the hour that objects have been detected and compares them with the request hour, as step 3 in Figure 5.3. Therefore, when the robot receives a request, it goes to the warmest spot in the heat map,

i.e., region of the environment where is more likely to find the target object, as step 4 in Figure 5.3. Lastly, if the requesting mode is executed before the recording one, and no data from the environment is available, our OS system would perform a brute force search since no information is available to improve its performance.

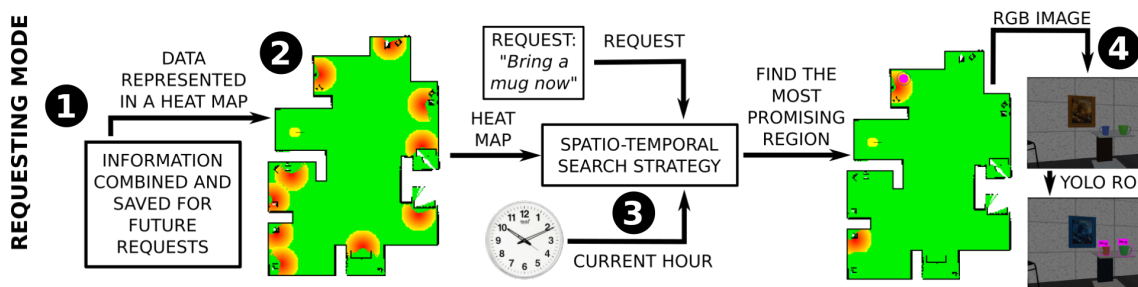


Figure 5.3 – Flowchart that explains how the requesting mode from our LSOS system works. The mode is the one that is executed when the robot is active with finding an object.

Although our LSOS system makes predictions based on the arrangement of objects, it does not have a separate training phase to learn the objects' presence, like machine learning algorithms. In fact, both modes (recording and requesting) can run in parallel, albeit the more information the map contains prior to the search, the better the search results tend to be. Another important point is that the semantic map in which the searches are based is not a black box. On the contrary, it relies on a probabilistic approach that infers the most promising regions to find the target object by recording how people interact with it. Besides, as the experiments presented below suggest, a small amount of data is enough for our system to succeed.

This section details our OS system and how it works. It starts with the description of the heat map module in both Sections 5.2.1 and 5.2.2. These sections explain how our LSOS system builds the heat map based on the data gathered by the recording mode. Finally, Section 5.2.3 presents the goal computation, which explains how our OS system estimates the most promising map regions to find the target object, i.e., the warmest spots in the heat map. It also discusses how our OS system behaves when the object is not found at the most promising region of the map.

5.2.1 The Heat Map and the representation of the objects' presence

The heat map is a visual technique widely used for visualizing complex spatial patterns, proposed by Kinney (KINNEY, 1993). It provides a meaningful and straightforward understanding for humans and processing programs. In this chapter, the discrete

distribution of objects' presence is processed into a continuous color distribution, in which the most likely regions to find the target object are intuitively revealed.

The 2D grid map \mathbf{m} that has been built by the SLAM system during the recording mode is used for computing the heat map \mathbf{h} . Both maps are equal in terms of the number of cells and size, but the difference is that a cell $\mathbf{m}_i \in \mathbf{m}$ is either free, occupied, or unknown, whereas the same cell $\mathbf{h}_i \in \mathbf{h}$ is the heat value that represents how likely that location is to contain a certain object. A cell in both \mathbf{m}_i and \mathbf{h}_i represents exactly the same place in the environment. Hence, only the cells that are either free or occupied in \mathbf{m} are likely to have a heat value different than zero in \mathbf{h} , as there are no objects on unknown regions in \mathbf{m} .

All the n objects that have been detected in the recording mode are represented by the set \mathbf{O} , in which $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_1, \dots, \mathbf{o}_n\}$. Each $\mathbf{o}_j \in \mathbf{O}$ is composed by its position within \mathbf{h} , its class, the hour it has been detected, and the robot's position during the detection, i.e., $\mathbf{o}_j = (o_j^p, o_j^c, o_j^h, o_j^r)$, respectively. To compute \mathbf{h} , we represent the presence of the object $\mathbf{o}_j \in \mathbf{O}$ by a weighted circular kernel $K(\cdot)$ of radius r . Let \mathbf{T} be the set of cells within the area of $K(\cdot)$ for a given r , and $\mathbf{T} \in \mathbf{h}$. For all cells $\mathbf{h}_i \in \mathbf{T}$, $K(\cdot)$ is defined as

$$K(D(\mathbf{h}_i, o_j^p), o_j^h) = \begin{cases} W(rh, o_j^h)Q(\mathbf{h}_i)(1 - \frac{d}{r}), & \text{if } d \leq r \\ 0, & \text{otherwise} \end{cases} \quad (5.1)$$

where $D(\mathbf{h}_i, o_j^p)$ is the Manhattan distance from the current cell being measured, $\mathbf{h}_i \in \mathbf{T}$ to the centre of the kernel, $o_j^p \in \mathbf{h}$. The function $Q(\cdot)$ is defined as

$$Q(\mathbf{m}_i) = \begin{cases} 0, & \text{if } \mathbf{m}_i \text{ is unknown in } \mathbf{m} \\ 1, & \text{otherwise} \end{cases} \quad (5.2)$$

and it checks whether a cell \mathbf{m}_i is unknown in \mathbf{m} . The other function $W(rh, o_j^h)$ computes the difference between the hour the object has been detected, o_j^h , and the hour the robot is requested to perform the search, rh . This difference is then used to compute the weight factor of the object \mathbf{o}_j , since the smaller is the hour difference, the more important that object becomes to the search. This function is defined as

$$W(rh, o_j^h) = 1 - \frac{|rh - o_j^h|}{12} \quad (5.3)$$

Here is important to mention that we use the 24-hour notation. Hence, $W(rh, o_j^h)$ is equal to one when rh and o_j^h are equal, and it is zero when they are 12 hours apart from each other, i.e., the largest difference in hours between two different time stamps. For example, if our LSOS system detected an object two times, at 8:00 in position A and 14:00 in position B, and it is performing the search at 10:00, it will probably start searching the object in A, as 8:00 is closer to 10:00 than 14:00.

The first part of computing the heat map h is depicted in Figure 5.4. The outcome of Equation 5.1 is shown in Figure 5.4c, in which every object from \mathbf{O} is represented by the kernel $K(\cdot)$. It is important to notice that the warmest cells in h , Figure 5.4c, are at the objects' positions in Figure 5.4a.

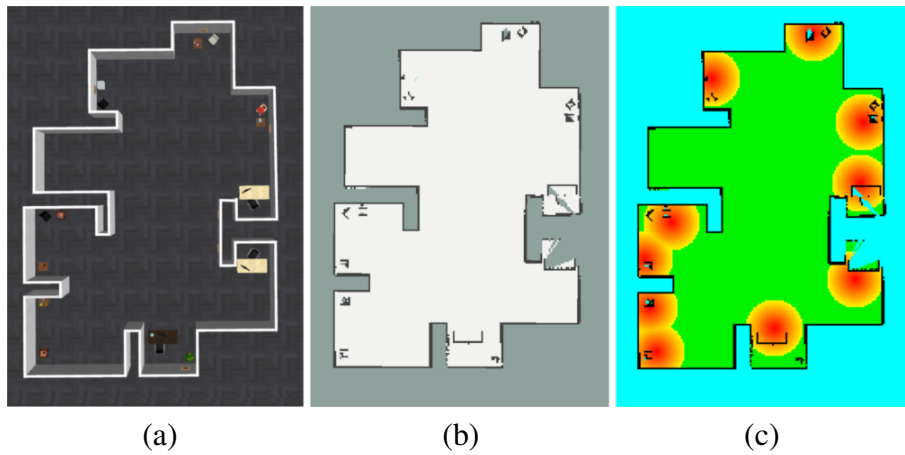


Figure 5.4 – Example of the computed heat map given a set of detected objects. (a) is the simulated environment with 10 objects spread in seven rooms. (b) is the 2D grid map built by Gmapping. (c) is the heap map built by our OS system, considering the map in (b) and the detected objects in (a). The warmer regions represent their position within the map.

The computed heat map h in Figure 5.4c indicates which regions our LSOS system should search, since it does not consider the colder spots (green region) in h . Besides, the class of an object helps to focus the searching in more promising regions, as there is no point in searching in a warm spot where there is only a book if our system is looking for a mug. Since the class of a detected object o_j is known, o_j^c , our system ignores warm spots from objects with different classes than the target one.

A third adjustment in the process of computing the heat map comes with a subtraction on the circular kernel's angle. During the search, our OS system guides the robot towards the edge of the most promising kernel. The edge, yellow cells in h , is one of the best regions to place the robot because it is the ideal distance between the robot and the kernel center (or the object's position). However, despite the ideal distance, positioning the robot at any place over the edge of the kernel does not ensure the object will be ei-

ther recognised or even within the camera's FoV. This issue is illustrated in Figure 5.5, in which the SR is quite close to the compute monitor at the edge of its kernel, but yet the object detection module is not able to detect it. Alternatively, the most appropriate position to place the SR would be the same one that the it has been in when recognizing the object during the recording mode. Thus, it is most likely that the object detection module will recognize the object once again if the robot assumes a similar pose it has assumed before. Hence, our heat map h is built considering the kernel $K(\cdot)$ in Equation 5.1 with an acute angle, instead of considering a 360° one. For a given object o_j , its kernel's angle is defined considering its robot's position when it has been recognised, i.e., o_j^r . Figure 5.6 illustrates the advantage of restricting the kernel's angle, in which the object detection module recognize the computer monitor.

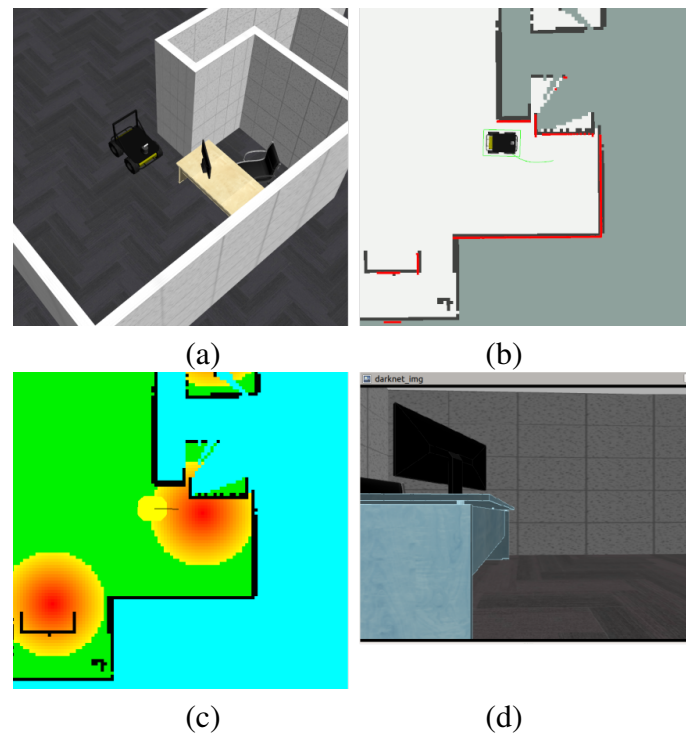


Figure 5.5 – Example of the robot at the edge of the kernel, and yet not recognising the object. (a) is the environment in simulation, (b) is the robot in m , (c) is the robot in h , and (d) the current image capture by the robot's camera.

The adjustments performed while computing the heat map h plays an important role in our LSOS system. Figure 5.7 represents the step by step of each adjustment, and the outcome is a few warmer spots in h . Besides saving computational resources by reducing the amount of regions to search, there is also an increase in the chances of detecting the target object by positioning SR in a better way.

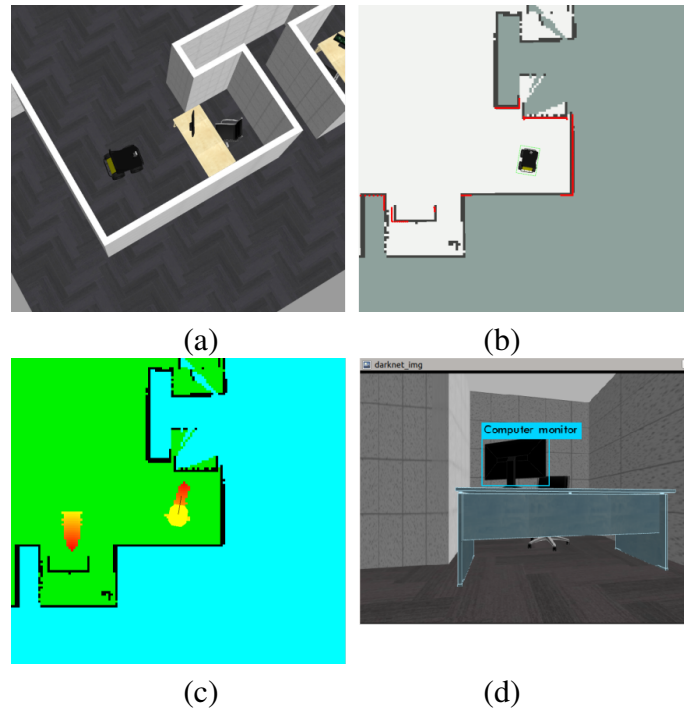


Figure 5.6 – Example of the modified kernel with an acute angle, and the recognised object. (a) is the environment in simulation, (b) is the robot in m , (c) is the robot in h , and (d) the current image capture by the robot's camera.

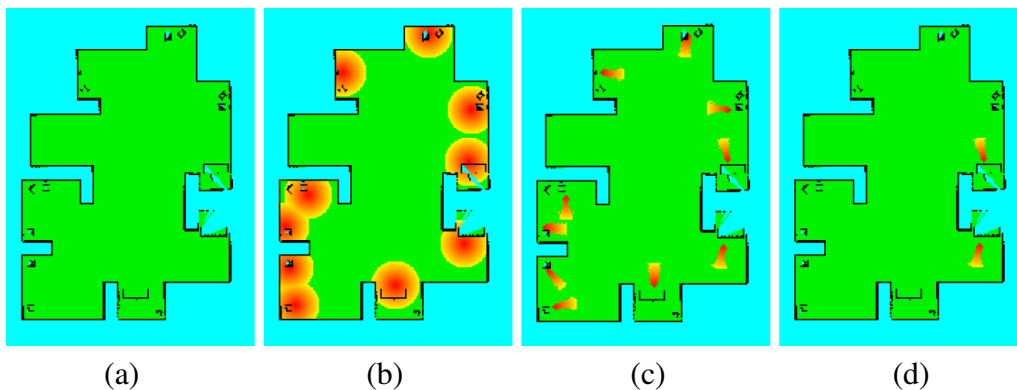


Figure 5.7 – Step by step of the space reduction performed by our OS system. (a) is the empty h , (b) is the representation of all objects from \mathbf{O} with the circular kernel $K(\cdot)$, (c) is the kernels' angle reduction given the robot's position of each object, and (d) is the outcome of considering only objects with the same class as the target's one and the other ones are ignored.

5.2.2 Inverted kernel $IK(\cdot)$

Our system performs another operation to make it easier to find the target object. In OS systems, one of the main goals is to place the robot at the most appropriate place to accomplish the task. For our LSOS system, as we already mentioned, the most promising regions to place the robot are at the kernels' edges due to the ideal distance to the objects. However, such edges in the kernel $K(\cdot)$ presented in Equation 5.1 are the coldest regions within the kernel area, as the warmest one is at the kernel's center. Hence, if our system

computes the inverse $IK(\cdot)$, the warmest regions becomes the ones at the edge, i.e., the ideal spots to place the robot during the search are the ones with the highest heat value. The $IK(\cdot)$ is similar to the one presented in Equation 5.1, and it is defined as

$$IK(D(\mathbf{h}_i, o_j^p), o_j^h) = \begin{cases} W(rh, o_j^h)Q(\mathbf{h}_i)\frac{d}{r}, & \text{if } d \leq r \\ 0, & \text{otherwise} \end{cases} \quad (5.4)$$

In summary, the difference between Equations 5.1 and 5.4 is that the former computes \mathbf{h} as the warmer regions being the object's position, whereas the latter computes \mathbf{h} as the warmer regions being the ideal place to position the robot during the search. Figure 5.8 shows their difference.

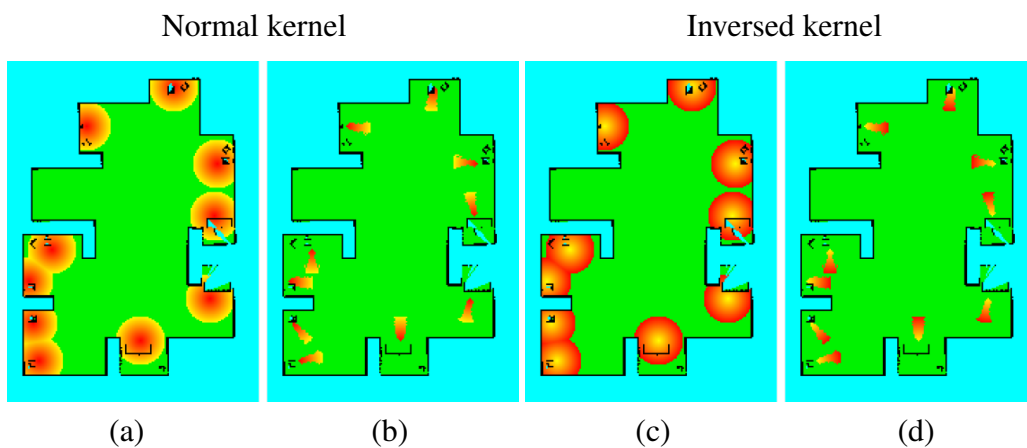


Figure 5.8 – The difference between the normal kernel and its inverse. (a) and (b) represent the normal kernel, the circular and reduced version, respectively. (c) and (d) represent the same for the inversed version of the same kernel.

Despite the easy readability by humans of \mathbf{h} built by Equation 5.1, due to the intuitive object's presence representation, for our OS system Equation 5.4 provides the most appropriate \mathbf{h} . Therefore, for now on, all references of \mathbf{h} within this work considers that it has been built using Equation 5.4.

5.2.3 Goal computation in the request mode

The process of building the heat map \mathbf{h} and reducing the regions to search for the object is the first part of our OS system, in which we represent the data from the recording mode in \mathbf{h} . In addition to that, it is also necessary to compute the goal, i.e., the most promising cell, \mathbf{h}_i^* , in \mathbf{h} that our system estimated to the robot go and find the target object. This computation is performed when the SR is requested to look for the target

object.

The goal computation is similar to the heat map construction because we also use a kernel approach. Our idea here is to analyse the whole \mathbf{h} in several circular kernels, and get the cell \mathbf{h}_i^* that neighbours have the highest probability of containing the target object, \mathbf{o}_j , for the request hour, rh . The final equation for computing the goal cell $\mathbf{h}_i^* \in \mathbf{h}$ is defined as

$$\mathbf{h}_i^* = \arg \max_{\mathbf{h}_i \in \mathbf{h}} (\varphi(\mathbf{h}_i, \mathbf{o}_j, rh)) \quad (5.5)$$

in which the function $\varphi(\mathbf{h}_i, \mathbf{o}_j, rh)$ computes the possibility of the cell \mathbf{h}_i contains the target object, \mathbf{o}_j . It is defined as

$$\varphi(\mathbf{h}_k, \mathbf{o}_j, rh) = \sum_{\mathbf{h}_i \in \mathbf{T}} H(\mathbf{h}_i, \mathbf{o}_j, rh) \quad (5.6)$$

where \mathbf{T} is a set of cells within the kernel area and $\mathbf{T} \subset \mathbf{h}$ centred in \mathbf{h}_k . The function $H(\cdot, \cdot, \cdot)$ returns the heat value (or probability) of the target object \mathbf{o}_j being in the cell \mathbf{h}_i , at the request hour, rh .

It is possible that for a certain request to our OS system, there are multiple regions in \mathbf{h} that register the presence of instances of the target object. Thus, our OS system computes Equation 5.5 over \mathbf{h} and sorts the multiple \mathbf{h}_i^* according their likelihood. The robot is moved towards the most likely one, and if the target object at that position is not found, the robot proceeds to the second most promising \mathbf{h}_i^* . This process is repeated until the robot visits all regions that register the presence of the target object. Lastly, in the worst case where the object was not found in any promising regions, the robot inspects all unvisited rooms, similar to a brute force search.

5.3 Experiments and Results

This section is divided into four subsections. Section 5.3.1 discusses the dataset and simulation setup used throughout the experiments. Section 5.3.2 explains the other two OS systems that we compare against our proposal, followed by Sections 5.3.3 and 5.3.4 that presents the experimental setup to which the three OS systems have been tested, along with their respective performances, respectively. Section 5.3.5 shows the performance of our LSOS system in estimating the person's presence given the HH106 dataset.

5.3.1 Simulation and dataset

Our proposal is to let the robot autonomously navigate through the environment until the target object's position. Such autonomy demands motion freedom that exists either in the real world or in a simulated environment that allows the robot to make decisions about its movement freely. Besides, as most of the OS systems in the literature are proposed assuming static environments, ignoring changes in the objects' position, the majority of the publicly available datasets are recorded in static environments. Unlike them, our LSOS system considers the semantic information of objects' position changes, and also uses other specific sorts of data, like the robot's pose. We assume that the environment is not static all the time. Hence, for testing and evaluating our system, we need a long-term dataset that provides information about the history of the objects' position, i.e., their location over time and what time they have been recognised at a specific location.

To the best of our knowledge, no dataset in the literature accomplishes our requirements, with all the high-level information our proposal needs. The datasets consist of a robot either moving through an environment many times but without objects' information or a single run and the class of the recognised object (HOWARD; ROY, 2003; LUO et al., 2007; AFIF et al., 2019). Besides, even though we could annotate the objects' position over time by watching the video from the robot's camera, we would not be able to move the objects as we want to mimic different patterns and routines, and we cannot ensure our annotations would be correct.

Therefore, we have used the Gazebo simulator to create an environment similar to an office with many small rooms connected, illustrated in Figure 5.9a. It measures 20.5 *m* by 14.5 *m* and contains seven rooms and objects in ten different places. We have chosen objects easily found in most offices, such as books, mugs, computers, and smartphones. Instances of different object classes are found in our simulated environment, at least in two different places, to test our system in scenarios with multiple promising places simultaneously. Creating our environment allows us to change the objects' position and even remove some of them. Thus, we can record the data using the recording mode with the objects' presence in different patterns, simulating multiple human routines. For this experiment, we use the Husky robot from Clearpath, Figure 5.9b. It is an unmanned ground vehicle equipped with a 2D LiDAR laser and an Intel RealSense RGB-D camera.

Despite the lack of public datasets appropriate for our LSOS system, we still would like to evaluate our OS system in a real scenarios. Therefore, we used a long-term

dataset where the target object is a human. The HH106 provides continuous ambient sensor data for human activity recognition, collected by 37 sensors spread over a nine-room apartment for two months (COOK, 2010). A single volunteer occupies the apartment, generating over 259,900 instances of data within the two months. We have considered only the motion sensors for our experiments, as they indicate where the volunteer is at a specific time. Figure 5.10 shows the apartment's floorplan and where the sensors have been placed for recording the data.

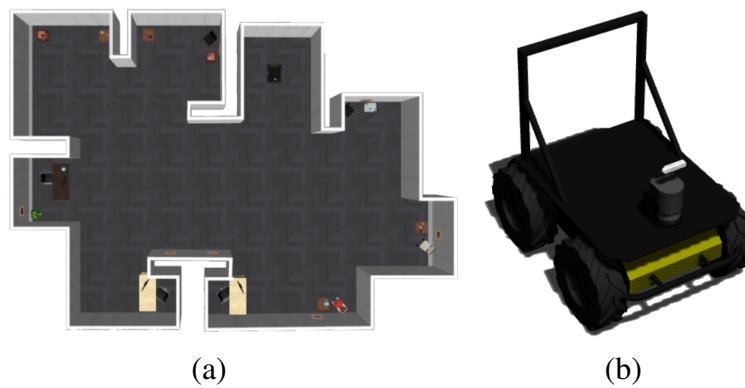


Figure 5.9 – (a) is the seven-rooms environment created on Gazebo simulator, and (b) is the Husky robot used throughout the experiments presented in this chapter.

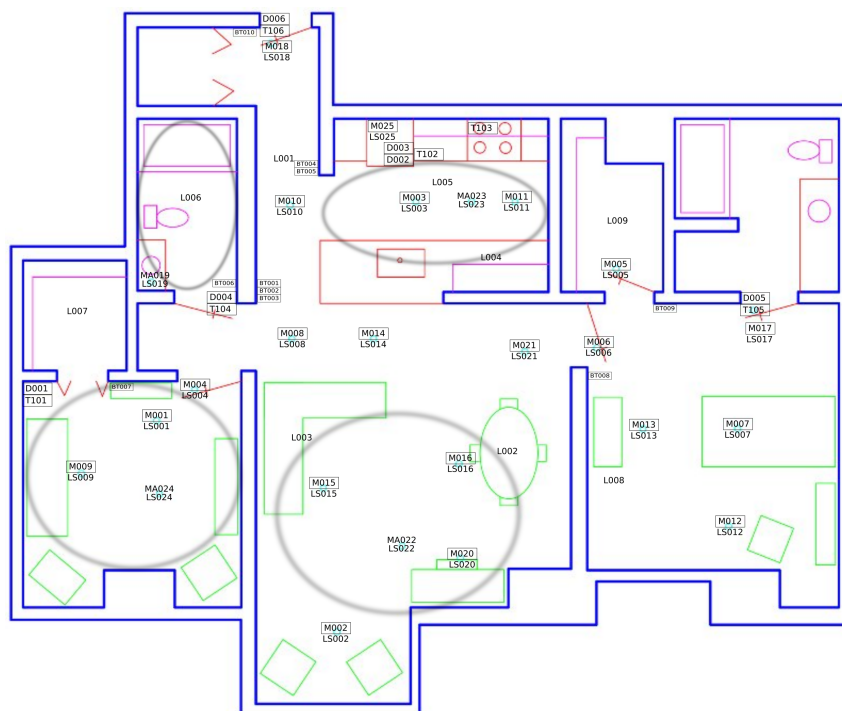


Figure 5.10 – The floorplan of the single-resident apartment from the HH106 dataset.

5.3.2 Two other object search systems to be compared against ours

We compare our LSOS system with a Brute Force and a Last Seen OS systems. They are briefly presented below as well as why they have been chosen to be tested in this chapter.

5.3.2.1 Brute Force OS system

Inspired by security patrol SRs that repeat the same route from time to time, this system makes the SR visit all locations according to a predefined route. Besides the robot's route, no information about the environment or the objects' presence is provided beforehand. In the field of OS, it is known as a brute force approach, which explains its name. Figure 5.11 depicts the clockwise route periodically repeated by the SR. Like any other brute force-based OS system, it does not ensure the robot will achieve the optimal performance in terms of the use of resources (battery and time), even though the robot finds the target object in most cases. In this chapter, such a system is the benchmark to be considered.

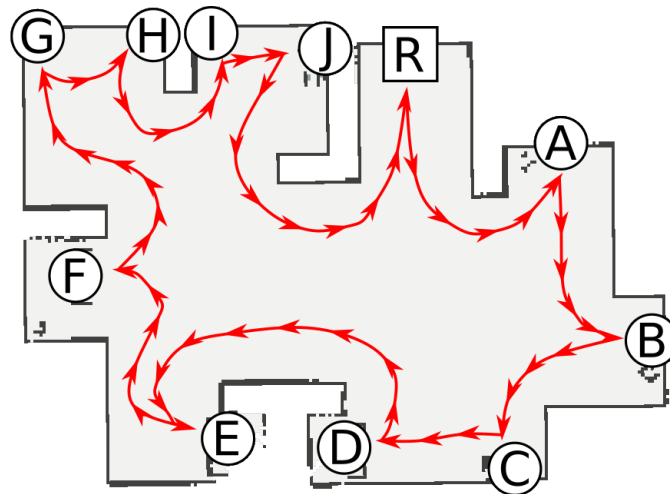


Figure 5.11 – The robot's route for the Brute Force OS system. The squared R is the robot's initial position, and then it follows the clockwise track in red from location A until location J, finishing back in R.

5.3.2.2 Last Seen OS system

This system is similar to our proposal in terms of the usage the semantic information about the organisation of objects over time. It guides the SR to the map spot where it has detected an instance of an object class that is equal to the target object, but only

the most recent object detection. The difference to our proposal is that it only considers the most recent observation, neglecting the rest of the objects' historic positions. Even though this system may save time and the robot's resources by guiding the robot straight to a specific spot, it does not ensure that the object will be found. The lack of information from the past may mislead the system, reducing its efficiency.

5.3.3 Simulated Experiment Setup

The three OS systems have been tested in different scenarios in simulation, with five objects' position patterns. Throughout the tests, we considered a simulated environment with objects being positioned in ten different locations, as can be seen in Figure 5.11. The systems have to find any instance of the target object, which is a mug. It is important to highlight that for this experiment we have chosen the mug, but it could be any other object class that YOLO (the object detection algorithm we are using) is able to detect. These instances could be in location A or H, Figure 5.11. We have chosen these locations because they are on opposite directions of the robot's initial position R in Figure 5.11. Hence, it highlights the efficiency of an OS system. If an OS system does not make the right decision since the beginning of the search, its performance will be inefficient.

The Table 5.1 depicts the data that is used in the experiments. To get the data, a robot repeated ten times a routine of visiting all rooms of our simulated environment while running the recording mode. Starting at 3:00, every routine was performed two hours apart, except for the period between 11:00 and 15:00. When an instance of mug was detected, our system recorded where it was (location A or H), and the hour of the detection. We could position instances of mug in many possible arrangements in the simulation, but this section does not intend to test all of them. Instead, we defined five different arrangements (patterns) for the objects, namely *Static*, *Static-Inv*, *Mobile*, *Mobile-Inv*, *Shift*. The five patterns in Table 5.1 have been manually designed as if the robot had recorded them on different days, and they are not related to each other. The experiments in this section aim to compare the three OS systems in different scenarios and investigate our OS system's performance in searches with little data available about the objects' position. Hence, when we test our OS system based on *Mobile*, for example, there are only its ten moments of the day that mugs have been detected to be considered for the estimations.

An instance of a mug has been detected both in locations A and H at different

Table 5.1 – Mug’s position recorded in five setups by the recording mode of our LSOS system.

Pattern	Recording hours									
	Morning					Evening				
	3:00	5:00	7:00	9:00	11:00	15:00	17:00	19:00	21:00	23:00
<i>Static</i>	A	A	A	A	A	H	H	H	H	H
<i>Static-Inv</i>	H	H	H	H	H	A	A	A	A	A
<i>Mobile</i>	A	H	A	H	A	H	A	H	A	H
<i>Mobile-Inv</i>	H	A	H	A	H	A	H	A	H	A
<i>Shift</i>	H	H	H	H	A	A	A	A	A	H

hours. On *Static*, the mug was in location A during the morning, from 3:00 until 11:00, and in location H during the evening, from 15:00 until 23:00. The opposite happened on *Static-Inv*, in which the mug was also moved between 11:00 and 15:00. On *Mobile* and *Mobile-Inv*, it is possible to note that the mug was moved more often, illustrating a different pattern of *Static* and *Static-Inv*. On *Shift*, the *Static-Inv* pattern is shifted back by one hour, and the mug is in a different location on the last hour of each period of the day (morning and afternoon).

For every setup in Table 5.1, the three OS systems were tasked with finding an instance of a mug at noon and midnight. These two requesting hours have been chosen because they are not within Table 5.1. Hence, the tested OS systems are not aware of the mug’s position at these times. The Table 5.2 shows the mug’s location for each requesting hour, and it is used as ground-truth for our experiments. We considered the mug remained at the position it was when detected the last time within each period of the day, i.e., at 11:00 and 23:00. Hence, the values of Table 5.2 are equal to the columns 11 and 23 of Table 5.1. It is important to highlight that the information in Table 5.2 is only used to compute the results, and this data is not provided to any of the OS systems tested.

Table 5.2 – Ground-truth of the mug’s position for every setup according to the requesting hours.

Pattern	Requesting hours	
	12:00	00:00
<i>Static</i>	A	H
<i>Static-Inv</i>	H	A
<i>Mobile</i>	A	H
<i>Mobile-Inv</i>	H	A
<i>Shift</i>	A	H

The Brute Force system repeated its route in every test, whereas the Last Seen considered the latest detection hour of every setup, which is at 23:00 for the five setups. Our proposed semantic OS system used all the data from each setup, e.g., for a search in *Static*, it uses only the data from its ten detections shown in Table 5.1. The robot’s traveled

distance measures the system's performance, i.e., the length of the robot's trajectory until either finding an instance of the target object or reporting that the object has not been found. Therefore, the shorter the traveled distance, the better is the system's performance.

5.3.4 Results and discussion from the simulated experiments

An OS system should efficiently find the target object, saving as much time and robot's battery as possible. This section presents and discusses the results of the three OS systems tested in this chapter. We measured their performance by computing the robot's traveled distance for searching the target object and whether they managed to find it. Every combination of OS system, requesting hour, and pattern was tested ten times, and from these tests, we computed the average and the standard deviation of the traveled distances, which are shown in Figure 5.12.

The *Static* and *Static-Inv* are characterized by only one change in the mug's position throughout the day, which happened at some time between 11:00 and 15:00. Their difference is that one is the inverted version of the other. In the results from both setups, Figures 5.12a and 5.12b, respectively, when the Last Seen OS system had to find an instance of mug at noon, it was the only one that failed. That is due to the mug's position at 23:00 in both setups. Their last mug detection in Table 5.1 happened at 23:00 in location H for the *Static*, and in location A for the *Static-Inv*. Hence, the Last Seen system wrongly guided the SR towards the opposite location in each setup. The Brute Force and our system found an instance of mug in both request searches for the two setups. However, there is a considerable difference in their performances for the request search at midnight in the *Static*, Figure 5.12a, and for the one at noon in the *Static-Inv*, Figure 5.12b. In such requests and setups, the mug was in the location H, as shown in Table 5.2, and the Brute Force system makes the SR travel a longer distance until it finds an instance of mug, as illustrated by the robot's trajectory in Figure 5.13d. In contrast, the Last Seen and our systems achieved the same goal traveling a distance three times shorter, like the examples in Figures 5.13e and 5.13f, respectively. For the request search at noon in the *Static*, and at midnight for the *Static-Inv*, the three systems have a similar result because the mug is in location A, Table 5.2, which is the first location the Brute Force guides the robot to, as shown in Figure 5.13a. Therefore, despite not reasoning over the available data, the Brute Force presents a similar result as ours.

Similar to the previous pair of patterns, the *Mobile* and *Mobile-Inv* represent an

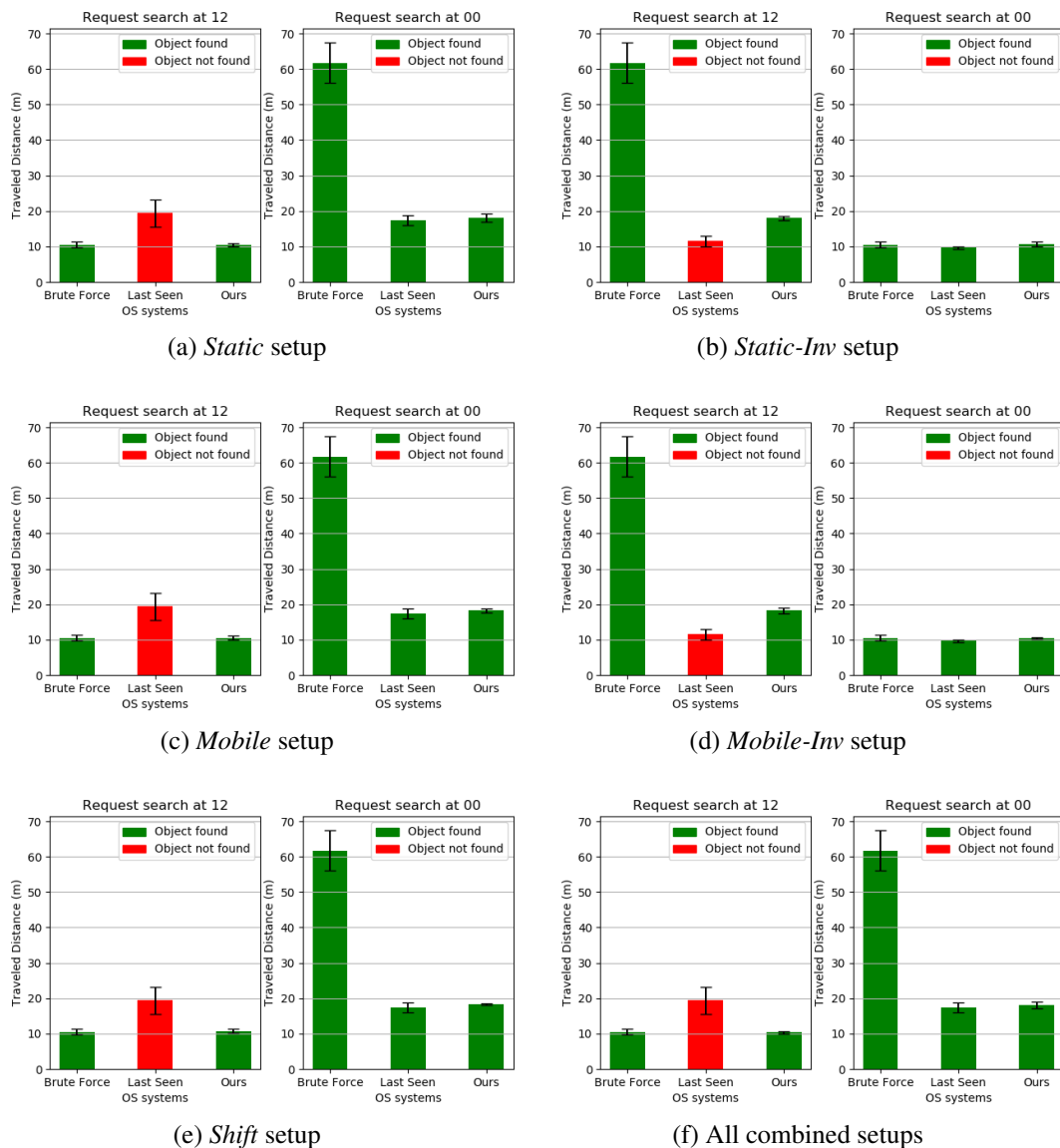


Figure 5.12 – Results of the three OS systems for both request search times considering many different setups.

arrangement in which the object is moved more often throughout the day. They aim to test the OS systems in more dynamic environments, in which the mug has constantly been moved. Figure 5.12c and Figure 5.12d show the results from the three OS systems in *Mobile* and *Mobile-Inv* patterns, respectively. In general, we see a similar outcome from the systems for the *Mobile* pattern pair than the one from the *Static* pair. The Brute Force always finds the mug, but as it just makes the SR repeats its route, it does not present a good performance when the mug is in location H, no matter the request search hour and the setup. Due to the same problem mentioned before, the Last Seen system is not able to consistently find the mug. In contrast to these two systems, our system is the only one that finds the target object traveling the shortest distance in most of the times. It is

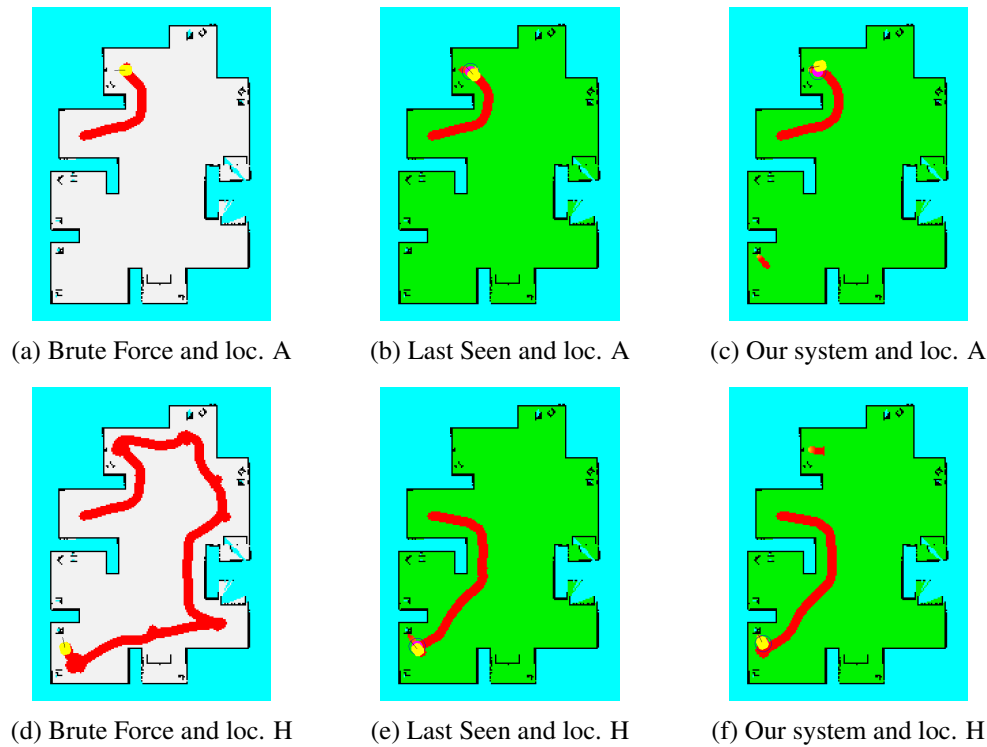


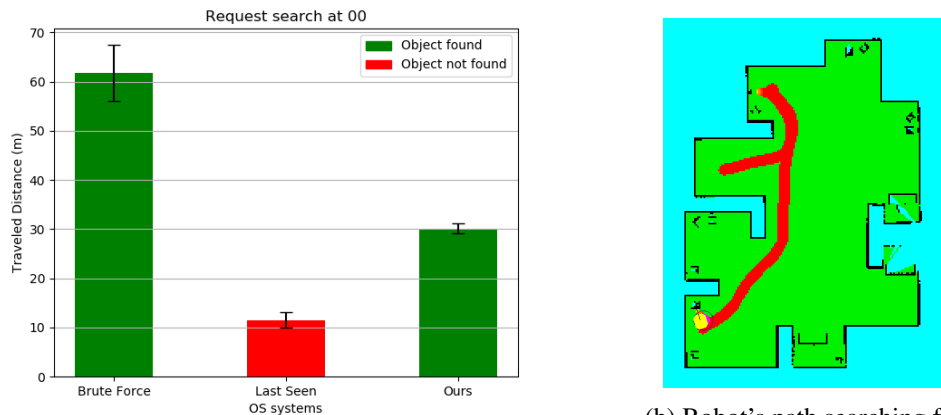
Figure 5.13 – Examples of the robot’s path for the search performed by the three OS systems in both locations A and H.

also worth mentioning that if our system does not find the target object with the smallest traveled distance for a search, its result is not significantly worse than the other systems. An example is shown in the requested search at midnight on *Mobile*, Figure 5.12c, where the Last Seen provided better results between the three systems.

It is important to compare the results from the Last Seen and our systems. Their difference is in the number of observations about the past object’s position that each considers. The Last Seen considers only the most recent one, whereas ours considers all of them. The Last Seen often fails to find the target object because relying on the newest object detection is unreliable. Hence, the robot ends up in a location that does not contain the target object, like demonstrated in Figure 5.15c. Our system shows the advantages of using all available data for more robust estimations, such as the results from the *Mobile* pattern pair. For the request search at noon on *Mobile*, our system estimates that it is more likely to find the mug in location A. It memorizes that the mug has been detected in location A three times, at 3:00, 7:00, and 11:00, and only two times in location H, at 5:00 and 9:00, Table 5.1. Besides the higher occurrence in location A, 11:00 is simply one hour before noon, whereas 9:00 is three, increasing A’s likelihood.

The results of the three OS systems for *Shift* are presented in Figure 5.12e. Our OS system has better performances than the other two systems, traveling the shortest distance

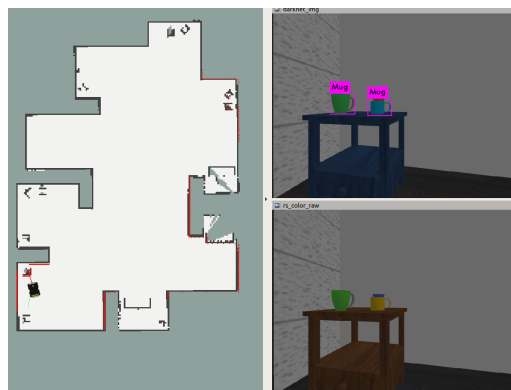
in all requested searches. It also presents the lowest standard deviation, meaning that our results are consistent throughout the ten runs.



(a) Results of the three OS systems

(b) Robot's path searching for the mug

Figure 5.14 – Results of the three OS systems for the request search at midnight in a modified version of *Mobile-Inv* in (a), and the robot's trajectory during the search generated by our OS system.



(a) Robot and mugs detected on location H



(b) Robot and mugs on location A

(c) Robot and no mug on location A

Figure 5.15 – Results of the three OS systems for both request search times considering many different setups.

There are two more experiments we carried out aiming to test our OS system's efficiency. In the first one, we submit our OS system to perform the object searching con-

sidering all data from the five setups together. We aim to measure the performance of our LSOS system in a scenario where the target object is moved in different patterns throughout a period of time, such as a week. The ground-truth for the setups combination is the same as the one from *Mobile-Inv*, and Figure 5.12f shows the results for this experiment.

In general, the results indicate that our system is efficient in finding the target object when moved in different patterns, as the target object was found in both request searches with the robot traveling the shortest distance between the three OS systems, as it is possible to observe comparing the Figures 5.13d and 5.13f. The Brute Force and the Last Seen systems presented similar results to the previous experiments since the pattern combination did not affect them. In the second experiment, we intentionally changed the mug's position from location A to location H on *Mobile-Inv* at midnight, Table 5.2. The goal is to analyse how our OS system behaves when the gathered data suggests the object is in a place, but it is somewhere else. Figure 5.14a illustrates the results, in which the Brute Force produces the longest traveled distance and the Last Seen failed in finding the mug. In contrast, our system accomplished the task, traveling a shorter distance than the Brute Force. Our system's estimation indicates location A as the first goal to find the mug. The estimation matches the data from Table 5.1, and since there is no information about the object's position at midnight, location A is considered as the most promising region to find the mug. As shown in Figure 5.14b, our system guides the robot to the location A. As soon as it does not find the target object in location A, it guided the robot towards the second most promising spot, location H, where an instance of mug was. Examples of our OS system in action are illustrated in Figure 5.15, in which the detected mugs and the robot's position on location H and location A are shown in Figure 5.15a, and in Figure 5.15b. The scenario that the mug was intentionally removed from location A is depicted in Figure 5.15c.

An extension to this experiment would be removing all instances of mug from environment. In this scenario, no even in location H our LSOS system would find a mug. Even though we have not presented this situation to our system, we would like to discuss the possible ways to overcome this problem. We argue that there are two main ideas: the first one would be finish the search when no object is found in the second possible location, and then report the outcome to the user. Depending on the further tasks the robot has to perform with the objects, this may not be suitable. The second idea is to make our LSOS system perform a brute force search and look for the target object in the unvisited regions. Since the rest of the environment are equally low likely

to contain the target object, our system could mimic the route performed by the Brute Force system and illustrated in Figure 5.11. For this second idea, the overall performance of our LSOS system may be worse than the Brute Force OS system's. It depends on the distance traveled by the robot while visiting all promising regions before figuring out that the object is not present in any of them. However, in contrast to the first idea that is not sure whether the object is in the environment, with this second idea our LSOS system could confirm that this information, and so the user can decide what to do next.

5.3.5 Person presence estimation with HH106 dataset

The HH106 is a long-term dataset, and its motion sensors provide the human location at different hours of the day in an apartment¹. Although the HH106 does not provide all information our OS system needs, such as the 2D map of the apartment, we adapted it to our context. Since there is no 2D map from the apartment, the robot cannot move and search for the human in the environment, and the heat map cannot be built. Hence, the traveled distance is not evaluated in this experiment. Instead, we only compute the human's presence, by the Equation 5.3, in every location of the apartment for the given request hour.

This experiment aims to evaluate our OS system in a large-scale dataset gathered in a real scenario. In this experiment, the sensor readings data from 59 days were provided to our OS system as if the recording mode gathered them. The data from the 60-th day of the dataset was used as the ground-truth to the experiments. Our LSOS system had to estimate the person's presence at specific hours in this last day that is unknown to it. We downsampled the dataset to represent the behavior of our system better as if the recording mode would have recorded the data hourly. Our system considers that the person is in the location that the person has spent most of the sixty minutes at a particular hour. Table 5.3 presents the downsampled dataset used by our OS system. Table 5.4 shows the person's presence at every hour of the day, except for the hours in which the sensors detected no motion.

Our LSOS system was requested to estimate the human's presence at five different hours for the 60-th day, which were 1:00, 10:00, 16:00, 18:00, and 22:00, as shown in Table 5.5. For this experiment, the higher is the score in Table 5.5, the more confident our OS system is about its estimation. In general, the estimations of our system

¹<http://casas.wsu.edu/datasets/>

Table 5.3 – The 59 days data from HH106 used by our OS system.

Daily Hours	Locations							
	W.A.	Liv.R.	Kit.	BedR.	Chair	Din.R.	BathR.	O.D.
00	0	2	1	21	0	1	3	2
01	2	1	3	20	0	1	1	2
02	3	5	1	29	0	0	0	0
03	1	4	3	24	0	0	0	2
04	3	0	1	28	0	2	0	1
05	0	1	2	35	0	1	0	1
06	1	3	3	31	0	1	4	10
07	1	4	4	10	0	6	5	7
08	3	2	4	4	1	14	8	5
09	6	2	1	1	0	15	3	12
10	3	1	1	3	0	3	11	19
11	6	3	7	2	0	3	5	13
12	7	3	2	1	1	14	4	12
13	5	8	3	0	5	4	6	13
14	4	5	1	0	2	5	4	16
15	9	4	5	0	0	4	2	17
16	16	5	4	1	2	6	2	7
17	16	2	4	2	0	9	1	7
18	21	4	3	1	0	3	2	13
19	14	5	2	1	0	4	2	11
20	13	8	2	0	0	6	7	2
21	14	3	0	8	0	13	8	1
22	1	3	2	37	0	1	6	0
23	1	2	0	34	1	1	1	0
Total	150	80	59	293	12	117	85	173

to the requested hours correspond to the human presence pattern previously indicated in Table 5.4. The person is in the bedroom from 21:00 until 5:00, which matches our OS system estimations for the requested hours at 1:00 and 22:00. For both request hours, our estimations for the BedR are considerably higher than the other locations, which suggests our system is confident about the human's presence at those hours. The requests at 10:00 and 16:00 are also correct compared to the Table 5.4 at the same daily hours. It is important to highlight the estimations from the requested at 16:00 and 18:00. Both estimations match the ground-truth from Table 5.4, but the scores from O.D. at 16:00 and the W.A. at 18:00 are pretty close. The minor difference is explained by the person's movement from the O.D. to the W.A., from 15:00 until 18:00, shown in Table 5.3. This movement makes our OS system estimate that both locations are possible, but with a small difference to the correct one.

Table 5.4 – The data from the last day of HH106 used to test our OS system.

Daily Hours	Locations							
	W.A.	Liv.R.	Kit.	BedR.	Chair	Din.R.	BathR.	O.D.
01	0	0	0	1	0	0	0	0
02	0	0	0	1	0	0	0	0
04	0	0	0	1	0	0	0	0
05	0	0	0	1	0	0	0	0
06	0	0	0	0	0	0	0	1
08	0	0	1	0	0	0	0	0
09	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	1	0
12	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	1
14	0	1	0	0	0	0	0	0
15	1	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	1
18	1	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	1
21	0	0	0	1	0	0	0	0
22	0	0	0	1	0	0	0	0
23	0	0	0	1	0	0	0	0
Total	4	1	1	7	0	1	1	5

Table 5.5 – The human’s presence estimation from our OS system.

Req. Hours	Locations							
	W.A.	Liv.R.	Kit.	BedR.	Chair	Din.R.	BathR.	O.D.
01	59.92	35.42	24.92	219.66	2.0	43.5	34.92	51.25
10	67.42	38.58	33.83	106.08	7.83	69.58	48.33	115.00
16	106.41	46.58	32.5	76.25	8.83	68.92	46.66	109.66
18	108.41	46.08	29.66	101.25	7.16	61.75	42.66	93.50
22	82.58	41.42	25.16	186.92	4.16	47.41	36.66	58.00

5.4 Summary

This chapter presented our LSOS system that uses organisational semantic information to estimate the target object’s position. The main contributions of this chapter are:

- a heat map that represents the objects’ presence, highlighting the most promising regions to position the robot, and then find an instance of the target object.
- a reduction in the number of regions to perform the search by the kernel’s angle contraction, which provides a better placement for the robot while searching.

- an OS system that observes the changes in the objects arrangement throughout a period of time, and then uses the organisational semantic information to estimate which regions of the environment are more likely to find the target object.
- an OS system for indoor environments that does not depend on a specific SLAM system and object-detection algorithm, and that can be executed alongside any other robotics application.
- a detailed analysis of the advantages of using the organisational semantic information of how the objects are moved within the environment during a certain period of time. Besides, the analysis also shows how it can save the robot's resources by making it travel shorter distances while searching.

Our proposed system was evaluated in simulation many times against two other OS systems, considering different patterns of the objects' positions and tested on the HH106 dataset gathered over two months. The experiments of our OS system with the HH106 dataset reveal that it performs well with long-term data, such as the two months of motion sensor readings. Besides, the other experiments with the simulated patterns of object's positions also demonstrate that our OS system succeeded in the search task with limited data of only ten instances within a day. Therefore, our system does not require extensive data about the object's position to accomplish the task.

Additionally, another important point highlighted by the experiments is that both the Brute Force and our OS systems are the only ones that find the target object under any circumstances. Although the Last Seen system travels a short distance in general, it sometimes fails in accomplishing the searching task. Despite the Brute Force's success in the searching task, it makes the robot spend way more resources than our OS system. Depending on the target object's location, the SR visited almost the entire environment until finding it, which contrasts with the results from our proposal. It is also worth mentioning that the larger is the environment, the more significant is the difference in performance between our system and the Brute Force since the robot's route in Brute Force also increases.

6 APPLICATION OF THE PROPOSED APPROACHES

This section describes how our approaches proposed in Chapters 4 and 5 could be deployed in an SR for disinfection tasks. Jaci is an SR built to help the fight against many bacterias and viruses, including the SARS-CoV-2 (MANTELLI et al., 2022). However, despite its efficiency in disinfecting indoor environments and the ability to explore the unknown environment room autonomously, it is not fully autonomous, and, in some cases, its disinfection process is time-consuming. First, Jaci still requires human assistance to find the room and position it there before disinfection. Second, it either disinfects the whole room or does not disinfect it at all. Although we have not been able to deploy our contributions presented in this thesis to Jaci, we believe they could enhance the robot's autonomy. Hence, Jaci would become an even more complete and fully autonomous SR.

Before explaining how our contributions can be adapted to the disinfection task carried out by Jaci, in Section 6.1 we introduce the SR Jaci and how it performs the disinfection. Next, in Section 6.2 we explain how semantic information about the organisation of the environment can improve Jaci's performance. In detail, we show how the robot can become fully autonomous by guiding itself to the rooms to perform the disinfection, Section 6.2.1, and how it can partially disinfect the environment under certain circumstances, Section 6.2.2. We conclude this chapter with further discussions in Section 6.3.

6.1 Jaci against COVID-19 and hospital infections

In the past few years, the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) has quickly and widely infected several countries, globally threatening human lives (ZHANG et al., 2020). The high infection rate of SARS-CoV-2 is associated but not exclusively with the virus's characteristic of remaining viable on surfaces for days, as demonstrated by studies (KAMPF et al., 2020; DOREMALEN et al., 2020). Besides, the environment disinfection has been recommended by researchers after they have found the SARS-CoV-2 present on surfaces of hospitals, threatening patients with COVID-19 (WU et al., 2020a; ONG et al., 2020). In addition, hospitals suffer with bacterial and fungus contamination, leading to high rates of infections. Thus, besides the recommended practices to reduce the spreading of the virus, like facial masks and social distancing, it would be very appropriate to use an effective tool to disinfect environments.

According to some studies provided by the research community, ultraviolet type-C

(UV-C) irradiation has produced a significant reduction in the incidence of microorganisms (ANDERSON et al., 2017; MARRA; SCHWEIZER; EDMOND, 2018). In addition to being a no-touch disinfection method, the UV lights have a wide range of incidence that quickly sanitises the air and nearby surfaces, save water, and are reusable. The combination of all these characteristics make the UV lights a suitable candidate for environment disinfection, and the increasing deployment of these lights in hospital and other health centres supports their effectiveness (MANTELLI et al., 2022). In contrast to these convenient characteristics of UV lights, prolonged exposure to them is not safe for humans due to the damage they can cause to the skin and eyes (KITAGAWA et al., 2021).

The SR Jaci comes into play to replace humans in UV-C disinfection tasks and prevent them from getting hurt (MANTELLI et al., 2022). Instor¹ has projected and built Jaci with eighteen UV-C lights attached to it in a two-layers tower shape, as shown in Figure 6.1a. Equipped with a 2D LiDAR and a few cameras (RGB and RGB-D), Jaci aims to navigate through the environment to perform the disinfection autonomously. It moves through free spaces within the environment and next to the border of obstacles, keeping an ideal distance from them and controlling its velocity to ensure a uniform disinfection.

Figure 6.1b illustrates Jaci disinfecting an operating room, circumventing the operating table and the surgical light. It is necessary to expose the existing microorganisms on obstacle surfaces to UV-C irradiation for a certain time to ensure their inactivation. An ideal UV dose must be delivered to a specific region to consider it sanitized (CHANPRAKON et al., 2019; CONTE; LEAMY; FURUKAWA, 2020). Jaci's autonomous system has been developed by Phi Robotics Research Lab² from the Federal University of Rio Grande do Sul, and it is responsible for both computing the disinfection trajectory and the delivered UV-C dose. While Jaci navigates through the free spaces of the environment, it also computes a dose map. This map estimates the amount of UV-C delivered on every part of the map. Hence, the constant update on the dose map helps the robot determine the regions that have already been sanitised and those that have not. Therefore, Jaci is a suitable disinfection tool that prevents humans from getting harmed by long-term UV exposure. Simultaneously, it reduces the spreading of the virus any other microorganism that may exist.

Although Jaci seems a complete solution to fight viruses and bacterias spreading, there is still room for improvement. Among the changes in Jaci's software, we high-

¹<<https://www.instor.com.br>>

²<<https://www.inf.ufrgs.br/phi-group/site>>

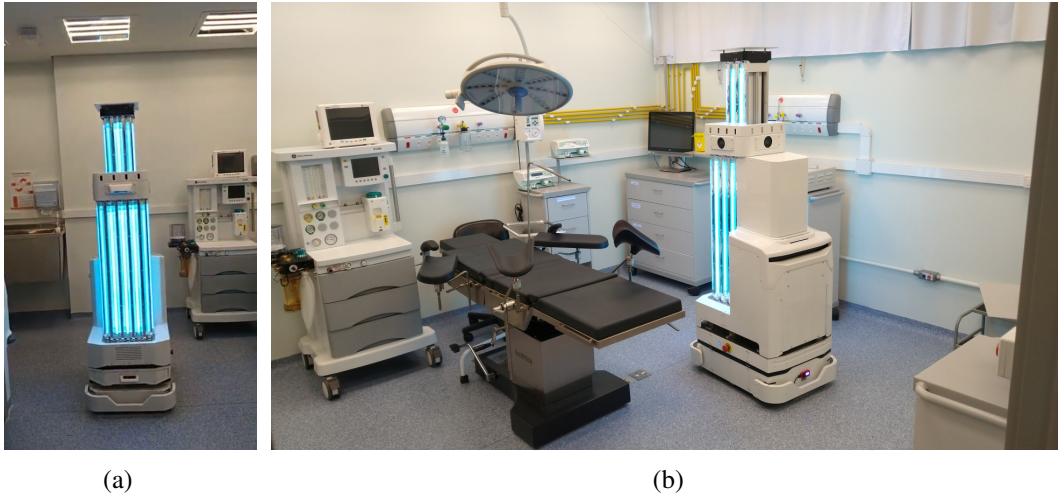


Figure 6.1 – Jaci is a disinfection robot developed by Instor, embedded with 18 UV-C lights in two different layers. (a) Jaci with the lights on, but not operating yet, and (b) Jaci in operation, disinfesting a hospital room.

light Jaci’s dependency on human aid to find and place it within the room to disinfect, and the lack of object-oriented disinfection. The first change is related to the preparation phase before starting the disinfection. Jaci relies on human assistance to be placed within the room to start the disinfecting process in its current version. On the other hand, the second change is associated with Jaci’s performance while performing the disinfection. Nowadays, Jaci only finishes the disinfection when the surface of objects in the free space boundary receives the ideal UV-C dose. It means that Jaci does not distinguish objects, i.e., regardless the objects there are in the environment, all of them will be disinfected. We believe that our contributions presented in this thesis can provide Jaci with the organisational semantic information necessary for these two improvements. Unfortunately, we have not been able to deploy our code to Jaci and carry out some experiments to evaluate the improvements in Jaci’s performance. Instor could not make it available for us to use it for a certain time to conduct the experiments, but we still plan in doing so in the near future. However, based on how our proposal works and the results presented in this thesis, we can discuss how they could be adapted for Jaci’s context and explain why they would be helpful.

6.2 Jaci and the environment organisation

Jaci is an SR that has been proposed to operate in indoor environments. Its efficiency in quickly sanitising environments, mainly due to the high amount of UV-C lights that combined deliver a high UV-C dose per second in a wide area in Jaci’s surround-

ings, allows the robot to be productive (MANTELLI et al., 2022). Depending on the arrangement of obstacles in the rooms, Jaci can disinfect many rooms before recharging its battery.

However, it is necessary to improve its autonomy by addressing the two issues mentioned above to maximise even more its efficiency. In our view, our contributions presented in this thesis, which relies on the organisational semantic information available in the environments where Jaci is supposed to operate, can provide the way for such efficiency improvement. Below we dive deep into how our systems could boost Jaci's efficiency.

6.2.1 Disinfecting a specific room

Imagine that Jaci is deployed in a large environment, where there are multiple rooms connected to corridors, like the hospital shown in Figure 6.2. Jaci has a preparation phase before starting its disinfection process in a given room in its present form. As briefly introduced in Section 6.1, it depends on human help to place it within the room, and then it is allowed to perform its task. This dependency is not an issue in small environments where there is just a few rooms to be disinfected, but it is a problem in places with a higher demand for the service provided by Jaci. Besides, this limitation worsens when a sequence of rooms must be sanitised, one right before the other. When Jaci finishes the disinfection in a certain room, it has to wait to be picked up by someone and then brought to the next room. The longer a waiting time between two disinfection processes, the longer it takes to disinfect the last room, and the more inefficient it becomes.

This limitation can be interpreted as an OS. Jaci has to find a particular room that must be disinfected, which is exactly the context of our contribution presented in Chapter 4. It could be imported to Jaci as an alternative to such an issue of human dependency. As shown in Chapter 4, our NSOS system guides a robot during a search for a specific door label in unknown environments. Analysing how the door labels are locally organised, our system estimates which regions are more promising to contain the target door label. Hence, the robot avoids the less likely regions and focuses on the most likely ones.

Jaci could autonomously search for the target door label in a human-out-of-the-loop process with our NSOS system. Instead of moving Jaci to the target door label, the person could request Jaci to find and disinfect it. Jaci may take more time to find the target than if a human would move it in the first few requests. However, as our NSOS system

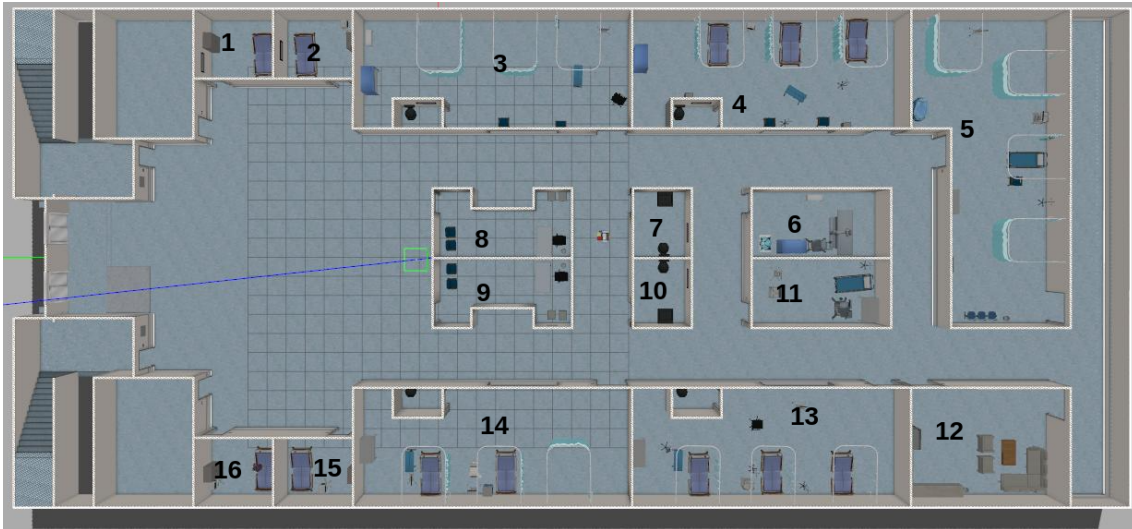


Figure 6.2 – Hospital environment used during in simulation during the development of Jaci’s autonomous system.

also builds the environment map while searching for the target, the more Jaci searches for different door labels, the more it explores (and maps) the environment. Consequently, it becomes more likely that Jaci does not need to search for every door label, as it discovers many door labels during a search. After a while, the whole environment would be mapped, which means that Jaci would be aware of all door labels. A further improvement could be using a path-planning approach for cases where the target door-label position is already known. Therefore, Jaci relies on a path-planning to compute the optimal path if the target door label has already been recognised and mapped. Otherwise, it relies on our NSOS system from Chapter 4 to search for the target.

Besides the hospital environment we mentioned before, there are other examples that highlight the improvement in the Jaci’s autonomy with our NSOS system. Places that remain closed for a few days in a row, like schools and offices during the weekends and holidays, are perfect scenarios for multiple rooms disinfected in sequence. This is because the lack of people within the buildings reduces the risk of someone getting hurt due to the UV light exposure, and makes all empty rooms ready to be disinfected. Then, with the aid of our NSOS system, a single person could set a list of rooms to be disinfected by Jaci during the days the place is empty.

6.2.2 Disinfecting a specific object within a room

Jaci’s autonomous system developed specifically for the sanitising task guides the robot throughout the free space, maintaining the robot to a certain distance from the ob-

stacle borders. The experiments with Jaci indicate that once the disinfection process has started, it only finishes when the whole room has received the ideal UV-C dose (except for the objects that are positioned in regions that Jaci's lights cannot reach) (MANTELLI et al., 2022). In summary, if Jaci can reach a certain region in the environment, it will be disinfected. Although the disinfection of the whole environment is ideal for most cases, Jaci could also offer the possibility of performing partial disinfection of certain objects in the room. For example, Jaci could disinfect the tables and computers more often than the bookshelves when operating in a library. Similarly, in a school, Jaci could disinfect the water dispensers in the hallways right before the break instead of disinfecting all the hallways (including all walls and other obstacles). By focusing on disinfecting only an object (or a list of objects), Jaci would be more effective and efficient in providing a safer and cleaner environment for people in short periods.

However, Jaci does not provide such a high-level disinfection option in its current version due to its purely geometric autonomous system. The map cells are represented only as free, occupied (obstacle), or unknown. Since this type of environment representation does not differentiate obstacles, there is no association between high-level concepts (or object classes) to some map areas. Consequently, Jaci's autonomous system for disinfection is unable to answer any other request other than "*disinfect the whole environment*". Besides lacking a high-level representation for objects, there is another issue with object-based disinfection. When Jaci is requested to disinfect a certain object, it needs to know where such a target object is to perform the disinfection. However, it is not appropriate to assume that the target object has already been mapped or that the object will be at the same position all the time. In case the target object has not been mapped or moved from its position, Jaci would have to search for it.

An alternative to overcome these issues is another contribution of our thesis presented in Chapter 5, the LSOS system. Besides providing the high-level representation of the environment, our LSOS system also performs the OS considering the object organisation over time. Combining our LSOS system with Jaci would allow the robot to efficiently search for the object when its position is unknown and perform the local disinfection. Another advantage would be scheduling the disinfection for more convenient periods for every place and situation. Since our LSOS system is temporal, it can estimate the most likely spot where the target object will be based on previous observations. Lastly, as it is very likely that the environment constantly needs to be disinfected, Jaci will have the opportunity of collecting more data about the objects for our LSOS system.

Hence, it is a mutual collaboration between Jaci and our LSOS system. Jaci gathers more data for the LSOS system while performing constant disinfection. Simultaneously, our LSOS system makes Jaci save resources by travelling short distances when searching for the target object.

6.3 Summary

The deployment of our contributions to Jaci would completely change how the user interacts with it. The current interaction is for a person to move it to the room and request it to start the disinfection process. However, with our NSOS system, the user could send a request to Jaci in a high-level form, e.g. “*disinfect the room number 12, and then the room number 15*”. The inclusion of our LSOS system would allow requests at an even higher level, such as “*disinfect the computer on room 12 at 11:00*”. The difference in the user’s request to Jaci reflects the improvement in terms of human-robot interaction and Jaci’s autonomy during the disinfection process. Although we were unable to carry out experiments with our contributions deployed to Jaci, their results presented in both Chapters 4 and 5 suggest the improvements that Jaci could achieve. The semantic information inferred from the organisation of the environment could bridge the gap between Jaci’s purely geometric autonomous disinfection system and its improved high-level version. Judging by the characteristics of this thesis contributions, they offer an ideal combination to boost Jaci’s performance, making it less human-dependent and easier for human-robot to interact.

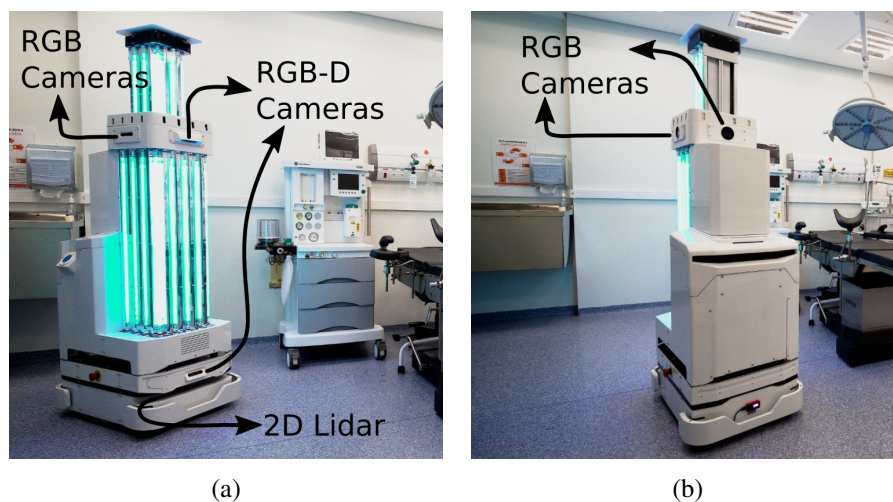


Figure 6.3 – Jaci’s sensor (a) in its front and left side, and (b) in its back and right side.

It is also important to analyse Jaci’s sensor kit, indicated in Figure 6.3, concerning

our contributions, i.e., is it necessary to make any changes in Jaci's hardware or software to make it compatible with our systems? The short answer is no, and we explain why. Jaci has a 2D LiDAR, which is used by the Gmapping SLAM system to build a 2D grid map m . This type of map is the same used throughout the works presented in this thesis, highlighting the harmony between current Jaci's mapping system and this thesis contributions. Besides, Jaci also has one RGB camera pointing on each of its sides, which could be used to read the door labels, similar to our setup shown in Figure 4.10. Lastly, there are two RGB-D cameras attached to Jaci, pointing forward. These two visual sensors provide a 3D point cloud each, which helps estimate the objects' position in relation to the robot, as we show in Figure 5.2. Therefore, Jaci's configuration in its current version supports the deployment of both our NSOS and LSOS systems, and no change in the hardware of Jaci is required.

7 CONCLUSION AND FUTURE WORK

In this thesis, we have explore the idea of semantic concepts to geometric entities in the robot's surroundings can be useful for mobile robotics in the context of high-level OS (CADENA et al., 2016). The use of semantics in mobile robotics aid to overcome the limitations of purely geometric SR's perception and maps. It aims to enhance robot's autonomy and robustness, as well as facilitate more complex tasks, like OS.

We have investigated the usage of semantic information associated with the spatial and temporal organisation of the environment in the context of OS problems. We argue that the organisational semantic information complements the robot's perception, expanding and improving it to aid in high-level tasks in everyday living spaces. The majority of the proposed solutions to the OS problems depend on geometric information only, such as the objects' 3D format or color, or even require a preparation step, like providing the association between rooms of the environments and possible objects that it may contain. The disadvantages of such proposals are that they are limited to the raw data read by the sensors or are not easy to deploy due to the preparation step they need before operating. This thesis goes beyond the geometric information and sensor readings, inferring organisational semantic information that complements the SR's geometric perception and enhances its performance in OS tasks.

We have devised two OS systems, NSOS and LSOS, presented in Chapters 4 and 5, respectively. They demonstrate that it is possible to model the organisation of both static and dynamic environments as semantic information, and use it for estimating objects' position. We have also demonstrated that these systems can rely on not trivial search clues, like the parity of numbers or the time an object has been detected, to efficiently deal with the OS problem. We present qualitative and quantitative results showing that the proposed OS systems have systematically outperformed the other OS systems tested in simulation.

In our NSOS system, we exploited the numbers from door signs to find a target door in a large and unknown environment. This strategy is inspired by how humans behave under the same circumstances, i.e., looking for a door sign in an unknown environment. Besides comparing each detected number of a door sign with the target one and check whether they are equal, humans also reason over the sequence of the door signs to estimate how the door signs are organised within the corridors. The search strategy of our NSOS system combines semantic information, namely the parity and growth properties

of the numbers, along with geometric information, the orientation of the corridor and the proximity between the robot and the promising regions, to estimate which region of the building is more favorable for the search. In case the robot is in a corridor that becomes less likely to contain the target door as the robot finds out more door signs, our OS system guides the robot towards another one more promising. For our NSOS system, a number from a door sign is associated to a door, which is the object that our system is actually aiming to find. Since the environment is entirely unknown to our system and no map is provided beforehand, it is doubtful that the SR will take the optimum path until the target door label while searching. Hence, more than guiding the robot straight towards the target's position, our OS system estimates when the current corridor is hopeless, and the SR should go elsewhere. The experiments in simulation and in real environment attest that our NSOS system always accomplishes the OS task, finding the target door.

In this thesis, we also managed to come up with a way to take advantage of the changes in the organisation of the environment through a period of time. It is no surprise that robots may operate in dynamic environments, and the task of searching for objects in these scenarios becomes even more challenging. This is the context of other contribution of our thesis, the LSOS system. It aims to explore the fact that the environment may change from time to time, and observing these changes in a long-term way may produce useful information. The search strategy of our LSOS system also works in a way that mimics humans. It assumes that objects are movable in real life, and it tries to understand how they are positioned over time to estimate their future position. The semantic part of this strategy processes the data the system gathers over a while. Then, it estimates the target object's position based on the history of positions of this same object. As the estimations of LSOS system being computed based on the collected data from the same environment, our system can adapt itself to the routine and habits of the person that interact with the objects. Hence, besides being generic and can be deployed in any environment, LSOS also adapts to the local singularities. Besides, the results suggest that the more consistent and repetitive the routine is, the more confident are the estimations. Lastly, the results also shows that the LSOS system can find the target object even when the object's arrangement varies considerably over a period of time.

The discussion about a possible deployment of both NSOS and LSOS systems to Jaci is worth to be mentioned. Besides showing in some high-level how they could be deployed to Jaci, it also explains the advantage of each system to this SR. Besides, the combination of both OS systems would bring to Jaci a considerable improvement in its

autonomy. In addition to increasing Jaci's efficiency in terms of the disinfected area in a certain time, the combination would also take humans out of the loop and make the disinfection process safer. The SR Jaci was not available for us to deploy our systems and carry out some experiments during the last few months. However, we still aim to do so as soon as Instor, the company that has built Jaci, gives us permission.

A drawback of our contributions is that they are too specific for their context, mainly the NSOS system. For example, our system is limited to scenarios where the door signs are design only with numbers. However, it was projected in a modular way, which allows the easy replacement of our current semantic planner to any other one with other heuristics. In the LSOS system, a drawback is the fact that it looks for instances of object classes. For cases in which the user wants an specific instace of object, e.g. a pair of Nike sneakers (and not just "shoes") or Don Quixote (and not just "book"), our LSOS system may finish the task finding any other instance of these examples. Nonetheless, this simplification is more associated to the limitation of the object detection algorithm, in our case YOLO, than to our LSOS system. Therefore, when the research community proposes an algorithm that is efficient in detecting such a fine level of details from objects, we could replace YOLO and provide a better search service.

There are still interesting new aspects of the object search that could be addressed. Outdoor environments also have great potential for SRs to perform searching tasks, considering that such scenarios present some sort of organisation. For example, most of the cities in the world have their houses numbered according to a certain rules, resulting in a large-scale organised environment. Even better, some cities also have a street naming system along with house numbering one, which could be combined into a coarse to fine approach (ASSOCIATION, 1950). Thus, if an SR is tasked with finding a specific house in these well-planned cities, its searching system could rely on the street naming rules to coarsely estimate the most likely streets to contain the target house. Next, the searching system could use the house numbering rules to find the target house. In this example, the SR would depend on a computer vision algorithm and a set of visual sensors to read the street names and house numbers. This may not be an issue nowadays with the advance of OCR algorithms and robustness of visual sensors. Besides, the research community has already presented promising results towards that functionality, like Oosterman and Green (2010) that used an iPhone to read the street names in New Zealand a few years ago.

Finally, using Ontology in the OS problem is also an interesting topic to consider. Ontology aims to describe a hierarchical structure composed by entities and relations for

purposes of representation (THOMAS, 2018). It shows the properties of a subject area, along with how they are related, by defining a set of concepts and categories that represent that subject. With the aid of Ontology to formally describe parts of outdoor environments, e.g. different stores, vehicles, and parks, an OS system could improve its performance. New clues could be used by the OS system, mainly the ones that specify the relation between properties of different objects.

REFERENCES

- ABUBAKAR, S. et al. Arna, a service robot for nursing assistance: System overview and user acceptability. In: IEEE. **2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)**. [S.l.], 2020. p. 1408–1414.
- AFIF, M. et al. A novel dataset for intelligent indoor object detection systems. **Artificial Intelligence Advances**, v. 1, 07 2019.
- ALMEIDA, A. T. de; FONG, J. Domestic service robots [tc spotlight]. **IEEE robotics & automation magazine**, IEEE, v. 18, n. 3, p. 18–20, 2011.
- AMHERST, U. **Room Numbering Guidelines**. [S.l.]: University of Massachusetts Amherst, 2012.
- ANDERSON, D. J. et al. Enhanced terminal room disinfection and acquisition and infection caused by multidrug-resistant organisms and clostridium difficile (the benefits of enhanced terminal room disinfection study): a cluster-randomised, multicentre, crossover study. **The Lancet**, Elsevier, v. 389, n. 10071, p. 805–814, 2017.
- ASSOCIATION, A. P. **Street Naming and House Numbering Systems**. [S.l.]: AMERICAN SOCIETY OF PLANNING OFFICIALS, 1950. <<https://www.planning.org/pas/reports/report13.htm>>. Accessed: 2022-05-09.
- AYDEMIR, A. et al. Plan-based object search and exploration using semantic spatial knowledge in the real world. In: **ECMR**. [S.l.: s.n.], 2011. p. 13–18.
- AYDEMIR, A.; JENSFELT, P. Exploiting and modeling local 3d structure for predicting object locations. In: IEEE. **International Conference on Intelligent Robots and Systems**. [S.l.], 2012. p. 3885–3892.
- AYDEMIR, A.; JENSFELT, P.; FOLKESSON, J. What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments. In: IEEE. **International Conference on Intelligent Robots and Systems**. [S.l.], 2012. p. 4675–4682.
- AYDEMIR, A. et al. Active visual object search in unknown environments using uncertain semantics. In: **Transactions on Robotics**. [S.l.]: IEEE, 2013. v. 29, n. 4, p. 986–1002.
- AYDEMIR, A. et al. Object search guided by semantic spatial knowledge. In: **The RSS**. [S.l.: s.n.], 2011. v. 11.
- AYDEMIR, A. et al. Search in the real world: Active visual object search based on spatial relations. In: IEEE. **International Conference on Robotics and Automation**. [S.l.], 2011. p. 2818–2824.
- BARBER, R. et al. Mobile robot navigation in indoor environments: Geometric, topological, and semantic navigation. In: **IntechOpen**. [S.l.: s.n.], 2018.
- BEGUM, M.; KARRAY, F. Visual attention for robotic cognition: a survey. In: **Transactions on Autonomous Mental Development**. [S.l.]: IEEE, 2010. v. 3, n. 1, p. 92–105.

BHATTACHARJEE, P.; GARG, A.; MITRA, P. Kago: an approximate adaptive grid-based outlier detection approach using kernel density estimate. **Pattern Analysis and Applications**, Springer, v. 24, n. 4, p. 1825–1846, 2021.

BJELONIC, M. **YOLO ROS: Real-Time Object Detection for ROS**. 2016–2018. <https://github.com/leggedrobotics/darknet_ros>.

BORENSTEIN, J.; KOREN, Y. Histogramic in-motion mapping for mobile robot obstacle avoidance. In: **Transactions on Robotics and Automation**. [S.l.: s.n.], 1991.

BORINATO, G. **Auto mowing system**. [S.l.]: Google Patents, 2017. US Patent 9,820,433.

BORJI, A. et al. Salient object detection: A survey. In: **Computational visual media**. [S.l.]: Springer, 2019. p. 1–34.

BURGARD, W. et al. Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach. In: IEEE. **Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)**. [S.l.], 1998. v. 2, p. 730–735.

CADENA, C. et al. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. In: **Transactions on Robotics**. [S.l.: s.n.], 2016.

CESAR, C. et al. Simultaneous localization and mapping: Present future and the robust-perception age. **arXiv preprint arXiv: 1606.05830**, 2016.

CHANPRAKON, P. et al. An ultra-violet sterilization robot for disinfection. p. 1–4, 2019.

CHEN, S.; LI, Y.; KWOK, N. M. Active vision in robotic systems: A survey of recent developments. In: **The International Journal of Robotics Research**. [S.l.: s.n.], 2011. v. 30, n. 11, p. 1343–1377.

CHEN, X.; LEE, S. Visual search of an object in cluttered environments for robotic errand service. In: **2013 IEEE International Conference on Systems, Man, and Cybernetics**. [S.l.: s.n.], 2013. p. 4060–4065.

CHIANG, A.-H.; TRIMI, S. Impacts of service robots on service quality. **Service Business**, Springer, v. 14, n. 3, p. 439–459, 2020.

CHUNG, T. H.; HOLLINGER, G. A.; ISLER, V. Search and pursuit-evasion in mobile robotics. In: **Autonomous robots**. [S.l.]: Springer, 2011. v. 31, n. 4, p. 299.

CONTE, D.; LEAMY, S.; FURUKAWA, T. Design and map-based teleoperation of a robot for disinfection of covid-19 in complex indoor environments. In: IEEE. **2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)**. [S.l.], 2020. p. 276–282.

COOK, D. J. Learning setting-generalized activity models for smart spaces. **IEEE intelligent systems**, NIH Public Access, v. 2010, n. 99, p. 1, 2010.

DAI, Z. et al. Transformer-xl: Attentive language models beyond a fixed-length context. **arXiv preprint arXiv:1901.02860**, 2019.

- DELLAERT, F. et al. Monte carlo localization for mobile robots. In: **IEEE. Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)**. [S.l.], 1999. v. 2, p. 1322–1328.
- DEVROYE, L. Nonparametric density estimation. **The L₁ View**, John Wiley, 1985.
- DEVROYE, L.; KRZYŻAK, A. On the hilbert kernel density estimate. **Statistics & probability letters**, Elsevier, v. 44, n. 3, p. 299–308, 1999.
- DEVROYE, L.; LUGOSI, G. **Combinatorial methods in density estimation**. [S.l.]: Springer Science & Business Media, 2001.
- DIVISION, P. **World Population Prospects**. New York, NY: Department of Economic and Social Affairs, United Nations, 2015.
- DOREMALEN, N. V. et al. Aerosol and surface stability of sars-cov-2 as compared with sars-cov-1. **New England journal of medicine**, Mass Medical Soc, v. 382, n. 16, p. 1564–1567, 2020.
- DOUCET, A. et al. Rao-blackwellised particle filtering for dynamic bayesian networks. **Conference on Uncertainty in Artificial Intelligence**, 2000.
- EKVALL, S.; KRAGIC, D.; JENSFELT, P. Object detection and mapping for service robot tasks. In: **Robotica**. [S.l.: s.n.], 2007. v. 25, n. 2, p. 175–187.
- ENGEL, J.; SCHÖPS, T.; CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In: **SPRINGER. European conference on computer vision**. [S.l.], 2014. p. 834–849.
- EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. **International journal of computer vision**, Springer, v. 88, n. 2, p. 303–338, 2010.
- FORLIZZI, J.; DISALVO, C. Service robots in the domestic environment: a study of the roomba vacuum in the home. In: **Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction**. [S.l.: s.n.], 2006. p. 258–265.
- GARCIA-HARO, J. M. et al. Service robots in catering applications: A review and future challenges. **Electronics**, MDPI, v. 10, n. 1, p. 47, 2020.
- GERKEY, B. **Gmapping package: laser-based Simultaneous Localization and Mapping for ROS**. 2013–2020. <<http://wiki.ros.org/gmapping>>.
- GIRDHAR, Y.; WHITNEY, D.; DUDEK, G. Curiosity based exploration for learning terrain models. In: **International Conference on Robotics and Automation**. [S.l.]: IEEE, 2014.
- GONG, C. et al. Frage: Frequency-agnostic word representation. **Advances in neural information processing systems**, v. 31, 2018.
- GRETTON, A. et al. A kernel two-sample test. **The Journal of Machine Learning Research**, JMLR. org, v. 13, n. 1, p. 723–773, 2012.
- GRISSETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. **IEEE transactions on Robotics**, IEEE, v. 23, n. 1, p. 34–46, 2007.

GRISSETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. In: **Transactions on Robotics**. [S.l.: s.n.], 2007.

GRISSETTI, G.; STACHNISS, C.; BURGARD, W. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: IEEE. **Proceedings of the 2005 IEEE international conference on robotics and automation**. [S.l.], 2005. p. 2432–2437.

HAIDEGGER, T. et al. Applied ontologies and standards for service robots. **Robotics and Autonomous Systems**, Elsevier, v. 61, n. 11, p. 1215–1223, 2013.

HAINES, B. **A basic model for numbering your rooms and spaces**. [S.l.]: FM:Systems, 2014.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HE, X.; ZHAO, K.; CHU, X. Automl: A survey of the state-of-the-art. **Knowledge-Based Systems**, Elsevier, v. 212, p. 106622, 2021.

HERSH, M. Overcoming barriers and increasing independence—service robots for elderly and disabled people. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England, v. 12, n. 8, p. 114, 2015.

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

ISLAM, N.; ISLAM, Z.; NOOR, N. A survey on optical character recognition system. **arXiv preprint arXiv:1710.05703**, 2017.

JONES, M.; KAPPENMAN, R. On a class of kernel density estimate bandwidth selectors. **Scandinavian Journal of Statistics**, JSTOR, p. 337–349, 1992.

JORGE, V. A. M. **Enabling loop-closures and revisits in active SLAM techniques by using dynamic boundary conditions an local potential distortions**. Thesis (PhD) — Federal University of Rio Grande do Sul, 2017.

JORGE, V. A. M. et al. Ouroboros: Using potential field in unexplored regions to close loops. In: **ICRA**. [S.l.: s.n.], 2015.

JUNG, K.; KIM, K. I.; JAIN, A. K. Text information extraction in images and video: a survey. In: **Pattern Recognition**. [S.l.: s.n.], 2004.

KAMPF, G. et al. Persistence of coronaviruses on inanimate surfaces and their inactivation with biocidal agents. **Journal of hospital infection**, Elsevier, v. 104, n. 3, p. 246–251, 2020.

Cormac Kinney. **HEATMAPS**, United States Patent and Trademark Office, United States, **75263259**. 1993.

KITAGAWA, H. et al. Effectiveness of 222-nm ultraviolet light on disinfecting sars-cov-2 surface contamination. **American journal of infection control**, Elsevier, v. 49, n. 3, p. 299–301, 2021.

KOK, L.; BERDEN, C.; SADIRAJ, K. Costs and benefits of home care for the elderly versus residential care: a comparison using propensity scores. **The European journal of health economics**, Springer, v. 16, n. 2, p. 119–131, 2015.

KRAJNÍK, T. et al. Long-term topological localisation for service robots in dynamic environments using spectral maps. In: IEEE. **2014 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.], 2014. p. 4537–4542.

KRAJNÍK, T. et al. Where’s waldo at time t? using spatio-temporal models for mobile robot search. In: IEEE. **2015 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.], 2015. p. 2140–2146.

KRAJNÍK, T. et al. Chronorobotics: Representing the structure of time for service robots. In: **International Symposium on Computer Science and Intelligent Control**. New York, NY, USA: Association for Computing Machinery, 2020.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, 2012.

KUMAR, V. et al. Industrial, personal, and service robots. **DRAFT REPORT**, v. 41, 2005.

LEE, M. K. et al. The snackbot: documenting the design of a robot for long-term human-robot interaction. In: **Proceedings of the 4th ACM/IEEE international conference on Human robot interaction**. [S.l.: s.n.], 2009. p. 7–14.

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, 1991.

LI, A. Q. et al. A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings. In: **Autonomous Robots**. [S.l.: s.n.], 2016.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 740–755.

LIN, X.; CHEN, T. A qualitative approach for the elderly’s needs in service robots design. In: **Proceedings of the 2018 International Conference on Service Robotics Technologies**. [S.l.: s.n.], 2018. p. 67–72.

LITZENBERGER, G. Service robots. **International Federation of Robotics**, 2018. Accessed: 2022-03-23. Available from Internet: <<https://ifr.org/service-robots>>.

LIU, L. et al. Deep learning for generic object detection: A survey. **International journal of computer vision**, Springer, v. 128, n. 2, p. 261–318, 2020.

LIU, Q. et al. Extracting semantic information from visual data: A survey. **Robotics**, Multidisciplinary Digital Publishing Institute, v. 5, n. 1, p. 8, 2016.

LUO, J. et al. Incremental learning for place recognition in dynamic environments. In: IEEE. **2007 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.], 2007. p. 721–728.

- MAFFEI, R. et al. Long-term place recognition using multi-level words of spatial densities. In: IEEE. **2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2016. p. 3269–3274.
- MAFFEI, R. et al. Integrated exploration using time-based potential rails. In: **International Conference on Robotics and Automation**. [S.l.]: IEEE, 2014.
- MAFFEI, R. et al. Fast monte carlo localization using spatial density information. In: **International Conference on Robotics and Automation**. [S.l.: s.n.], 2015.
- MAFFEI, R. et al. Using n-grams of spatial densities to construct maps. In: **International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2015.
- MAFFEI, R. d. Q. **Translating sensor measurements into texts for localization and mapping with mobile robots**. Thesis (PhD) — Federal University of Rio Grande do Sul, 2017.
- MANTELLI, M. et al. Semantic temporal object search system based on heat maps (*under review*). **Journal of intelligent & robotic systems**, Springer, 2022.
- MANTELLI, M. et al. Semantic active visual search system based on text information for large and unknown environments. **Journal of intelligent & robotic systems**, Springer, v. 101, n. 2, p. 1–23, 2021.
- MANTELLI, M. F. et al. Autonomous environment disinfection based on dynamic uv-c irradiation map. **IEEE Robotics and Automation Letters**, IEEE, 2022.
- MARRA, A. R.; SCHWEIZER, M. L.; EDMOND, M. B. No-touch disinfection methods to decrease multidrug-resistant organism infections: a systematic review and meta-analysis. **infection control & hospital epidemiology**, Cambridge University Press, v. 39, n. 1, p. 20–31, 2018.
- MARTINIC, G. The proliferation, diversity and utility of ground-based robotic technologies. **Canadian Military Journal**, v. 14, n. 4, p. 52, 2014.
- MCKIM. **Hotel - Pennsylvania typical floor plan**. 1919. <<https://bit.ly/36Y6t5P>>.
- MUANDET, K. et al. Kernel mean embedding of distributions: A review and beyond. **Foundations and Trends® in Machine Learning**, Now Publishers, Inc., v. 10, n. 1-2, p. 1–141, 2017.
- MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. **IEEE transactions on robotics**, IEEE, v. 31, n. 5, p. 1147–1163, 2015.
- MURPHY, K. P. et al. Bayesian map learning in dynamic environments. In: **NIPS**. [S.l.: s.n.], 1999. p. 1015–1021.
- NARAYANA, M. et al. Lifelong update of semantic maps in dynamic environments. In: IEEE. **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2020. p. 6164–6171.
- NEUMANN, L.; MATAS, J. Real-time scene text localization and recognition. In: **Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2012.

- OKSANEN, J. et al. Methods for deriving and calibrating privacy-preserving heat maps from mobile sports tracking application data. **Journal of Transport Geography**, Elsevier, v. 48, p. 135–144, 2015.
- ONG, S. W. X. et al. Air, surface environmental, and personal protective equipment contamination by severe acute respiratory syndrome coronavirus 2 (sars-cov-2) from a symptomatic patient. **Jama**, American Medical Association, v. 323, n. 16, p. 1610–1612, 2020.
- OOSTERMAN, J.; GREEN, R. Geolocation on the iphone by automatic street sign reading. In: IEEE. **2010 25th International Conference of Image and Vision Computing New Zealand**. [S.l.], 2010. p. 1–6.
- PANGERCIC, D. et al. Semantic object maps for robotic housework-representation, acquisition and use. In: IEEE. **2012 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.], 2012. p. 4644–4651.
- PAULIUS, D.; SUN, Y. A survey of knowledge representation in service robotics. **Robotics and Autonomous Systems**, Elsevier, v. 118, p. 13–30, 2019.
- PRESTES, E.; ENGEL, P. M. Exploration driven by local potential distortions. In: **2011 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2011. p. 1122–1127.
- PRESTES, E. et al. Exploration method using harmonic functions. In: **Robotics and Autonomous Systems**. [S.l.: s.n.], 2002.
- RASOULI, A. et al. Attention-based active visual search for mobile robots. In: **Autonomous Robots**. [S.l.]: Springer, 2020. v. 44, n. 2, p. 131–146.
- RASOULI, A.; TSOTSOS, J. K. Integrating three mechanisms of visual attention for active visual search. In: **arXiv preprint arXiv:1702.04292**. [S.l.: s.n.], 2017.
- REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.
- ROBOTICS, I. . **ISO 8373:2012 Robots and robotic devices — Vocabulary**. International Organization for Standardization, 2012. Accessed: 2022-03-23. Available from Internet: <<https://www.iso.org/standard/55890.html>>.
- ROGERS, J. G.; CHRISTENSEN, H. I. Robot planning with a semantic map. In: **International Conference on Robotics and Automation**. [S.l.: s.n.], 2013.
- ROSETE, A. et al. Service robots in the hospitality industry: An exploratory literature review. In: SPRINGER. **International Conference on Exploring Services Science**. [S.l.], 2020. p. 174–186.
- RUSSELL, B. C. et al. Labelme: a database and web-based tool for image annotation. **International journal of computer vision**, Springer, v. 77, n. 1, p. 157–173, 2008.
- SAIDI, F.; STASSE, O.; YOKOI, K. Active visual search by a humanoid robot. In: **Robotics: Viable Robotic Service to Human**. [S.l.]: Springer, 2007. p. 171–184.

SALAS-MORENO, R. F. et al. Slam++: Simultaneous localisation and mapping at the level of objects. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2013. p. 1352–1359.

SCHÖLKOPF, B. et al. **Learning with kernels: support vector machines, regularization, optimization, and beyond**. [S.l.]: MIT press, 2002.

SCHULZ, R. et al. Robot navigation using human cues: A robot navigation system for symbolic goal-directed exploration. In: **International Conference on Robotics and Automation**. [S.l.: s.n.], 2015.

SCOTT, D. W. **Multivariate density estimation: theory, practice, and visualization**. [S.l.]: John Wiley & Sons, 2015.

SEDDIGH, M. et al. A comparative study of perceived social support and depression among elderly members of senior day centers, elderly residents in nursing homes, and elderly living at home. **Iranian Journal of Nursing and Midwifery Research**, Wolters Kluwer–Medknow Publications, v. 25, n. 2, p. 160, 2020.

SEIDITA, V. et al. Robots as intelligent assistants to face covid-19 pandemic. **Briefings in Bioinformatics**, Oxford University Press, v. 22, n. 2, p. 823–831, 2021.

SILVA, E. e et al. Bvp-exploration: further improvements. In: **Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)**. [S.l.: s.n.], 2003. v. 4, p. 3239–3244 vol.3.

SILVERMAN, B. W. **Density estimation for statistics and data analysis**. [S.l.]: Chapman and Hall, 1986.

SJöö, K.; AYDEMIR, A.; JENSFELT, P. Topological spatial relations for active visual search. In: **Robotics and Autonomous Systems**. [S.l.: s.n.], 2012. v. 60, n. 9. ISSN 0921-8890.

SJöö, K. et al. Object search and localization for an indoor mobile robot. In: **Journal of Computing and Information Technology**. [S.l.]: SRCE-University Computing Centre, 2009. v. 17, n. 1.

SPRUTE, D. et al. Ambient assisted robot object search. In: SPRINGER. **International Conference on Smart Homes and Health Telematics**. [S.l.], 2017. p. 112–123.

TALBOT, B. et al. Find my office: Navigating real space from semantic descriptions. In: **International Conference on Robotics and Automation**. [S.l.: s.n.], 2016.

THAMRONGAPHICHARTKUL, K. et al. A framework of iot platform for autonomous mobile robot in hospital logistics applications. In: IEEE. **2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)**. [S.l.], 2020. p. 1–6.

THEURER, K. et al. The need for a social revolution in residential care. **Journal of aging studies**, Elsevier, v. 35, p. 201–210, 2015.

THOMAS, C. **Ontology in Information Science**. [S.l.]: BoD–Books on Demand, 2018.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics**. [S.l.]: Massachusetts Institute of Technology, 2006.

TORRESEN, J.; KURAZUME, R.; PRESTES, E. Special issue on elderly care robotics – technology and ethics. **Journal of Intelligent & Robotic Systems**, Springer Nature BV, v. 98, n. 1, p. 3–4, 2020.

TORRESEN, J. et al. Robot companions for older people – ethical concerns. **International Conference on Robot Ethics and Standards**, p. 53, 2018.

TSOTSOS, J. K. On the relative complexity of active vs. passive visual search. In: **International journal of computer vision**. [S.l.]: Springer, 1992. v. 7, n. 2, p. 127–141.

UNIVERSITY, S. **Room Numbering Guidelines**. [S.l.]: Stanford University, 2017.

VAKILIAN, K. A.; MASSAH, J. A farmer-assistant robot for nitrogen fertilizing management of greenhouse crops. **Computers and electronics in agriculture**, Elsevier, v. 139, p. 153–163, 2017.

VASUDEVAN, S. et al. Cognitive maps for mobile robots—an object based approach. **Robotics and Autonomous Systems**, Elsevier, v. 55, n. 5, p. 359–371, 2007.

VEIGA, T. S. et al. Efficient object search for mobile robots in dynamic environments: Semantic map as an input for the decision maker. In: **International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2016.

VINTR, T. et al. Spatio-temporal representation for long-term anticipation of human presence in service robotics. In: IEEE. **2019 International Conference on Robotics and Automation (ICRA)**. [S.l.], 2019. p. 2620–2626.

WAN, A. Y. S. et al. Waiter robots conveying drinks. **Technologies**, Multidisciplinary Digital Publishing Institute, v. 8, n. 3, p. 44, 2020.

WANG, C. et al. Efficient object search with belief road map using mobile robot. **IEEE Robotics and Automation Letters**, IEEE, v. 3, n. 4, p. 3081–3088, 2018.

WU, S. et al. Environmental contamination by sars-cov-2 in a designated hospital for coronavirus disease 2019. **American journal of infection control**, Elsevier, v. 48, n. 8, p. 910–914, 2020.

WU, Z. et al. A comprehensive survey on graph neural networks. **IEEE transactions on neural networks and learning systems**, IEEE, v. 32, n. 1, p. 4–24, 2020.

YANG, G.-Z. et al. **Combating COVID-19—The role of robotics in managing public health and infectious diseases**. [S.l.]: American Association for the Advancement of Science, 2020. eabb5589 p.

YE, Q.; DOERMANN, D. Text detection and recognition in imagery: A survey. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 37, n. 7, p. 1480–1500, 2014.

YE, Y.; TSOTSOS, J. K. On the collaborative object search team: a formulation. In: **Distributed Artificial Intelligence Meets Machine Learning Learning in Multi-Agent Environments**. [S.l.]: Springer, 1996. p. 94–116.

YE, Y.; TSOTSOS, J. K. Sensor planning for 3d object search. **Computer Vision and Image Understanding**, Elsevier, v. 73, n. 2, p. 145–168, 1999.

YE, Y.; TSOTSOS, J. K. A complexity-level analysis of the sensor planning task for object search. In: **Computational Intelligence**. [S.l.: s.n.], 2001. v. 17, n. 4, p. 605–620.

ZENG, Z.; RÖFER, A.; JENKINS, O. C. Semantic linking maps for active visual object search. In: **arXiv preprint arXiv:2006.10807**. [S.l.: s.n.], 2020.

ZHANG, H. et al. Text extraction from natural scene image: A survey. In: **Neurocomputing**. [S.l.: s.n.], 2013.

ZHANG, J. et al. Challenges of sars-cov-2 and lessons learnt from sars in guangdong province, china. **Journal of Clinical Virology**, Elsevier, v. 126, p. 104341, 2020.

ZHANG, X. et al. Object class detection: A survey. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 46, n. 1, p. 1–53, 2013.

ZHANG, Z. et al. Multi-oriented text detection with fully convolutional networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 4159–4167.

ZHOU, T. et al. Cascaded human-object interaction recognition. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 4263–4272.

ZOU, Z. et al. Object detection in 20 years: A survey. **arXiv preprint arXiv:1905.05055**, 2019.

APPENDIX A — RESUMO EXPANDIDO

Robôs podem ser agrupados em diferentes classes dependendo de suas funções e do local de trabalho para o qual foram projetados (KUMAR et al., 2005; ROBOTICS, 2012; HAIDEGGER et al., 2013). Dentre todas as classes de robôs, os robôs de serviço (RSs), são robôs que trabalham de forma semi ou totalmente autônoma para realizar serviços úteis, excluindo operações de manufatura (ROBOTICS, 2012). Os RSs vêm em diferentes formatos, pois podem ou não ser equipados com uma estrutura de braço e, embora a maioria deles sejam móveis, eles também podem ser fixados no local (GARCIA-HARO et al., 2020). A Federação Internacional de Robótica (FIR) divide RSs em duas subclasses com base em sua usabilidade: *RS professional* e *personal/domestic* (LITZENBERGER, 2018). Alguns exemplos de RS profissional (RSPs) são robôs de defesa (MARTINIC, 2014), médicos (ABUBAKAR et al., 2020), logísticos (THAMRONGAPHICHARTKUL et al., 2020) e assistentes de fazendeiros (VAKILIAN; MASSAH, 2017). Exemplos de RS doméstico (RSDs) incluem, mas não estão limitados a aspiradores de pó (FORLIZZI; DISALVO, 2006), cortadores de grama (BORINATO, 2017), assistentes idosos (HERSH, 2015) e garçons de alimentos e bebidas (WAN et al., 2020). O mercado de RSs vem crescendo regularmente, e não é surpresa que haja uma expectativa de que ele cresça ainda mais nos próximos anos (ALMEIDA; FONG, 2011; CHIANG; TRIMI, 2020). O custo decrescente dos componentes de hardware (processadores, drivers de motor e sensores), o aumento da densidade de energia e o menor custo das baterias e as ameaças causadas por surtos como o COVID-19 impulsionam essa expansão (CHIANG; TRIMI, 2020).

De acordo com a Divisão de População (PD) das Nações Unidas, em 2015, haviam 901 milhões de pessoas com 60 anos ou mais, representando 12% da população global (DIVISION, 2015). Além disso, a PD projeta que, até 2030, o número de idosos no mundo chegará a 1,4 bilhão e 2,1 bilhões até 2050. Diversas políticas para enfrentar os problemas de envelhecimento populacional têm sido propostas por vários países, incluindo, por exemplo, asilos e infra-estrutura para os idosos (LIN; CHEN, 2018; SEDDIGH et al., 2020). No entanto, colocar idosos em casas de repouso ou até mesmo em asilos pode causar alguns problemas, como dependências físicas, emocionais e psicológicas (THEURER et al., 2015). Além disso, alguns idosos não permanecem voluntariamente em asilos, preferindo passar os anos restantes em suas casas, onde têm uma autoimagem mais positiva do que aqueles que moram em lares de idosos (KOK; BERDEN; SADIRAJ, 2015;

LIN; CHEN, 2018). O número crescente de idosos que vivem em casa apoia a necessidade de RSDs para automatizar processos e tarefas que podem ser tediosas, inconvenientes ou até mesmo desafiadoras para os idosos (PAULIUS; SUN, 2019; TORRESEN; KURAZUME; PRESTES, 2020). Em geral, esses tipos de robôs podem contribuir para tarefas práticas para humanos como robôs assistentes ou companheiros, tais como observar idosos em situações de emergências, lembrá-los de tomar seus remédios e procurar, e pegar e colocar objetos (SPRUTE et al., 2017; TORRESEN et al., 2018; PAULIUS; SUN, 2019).

Além disso, embora algumas das principais motivações para a implantação de RSs tenham sido a assistência aos idosos e a melhoria da produtividade, a pandemia do COVID-19 trouxe um propósito mais crítico para eles (CHIANG; TRIMI, 2020). Os RSPs podem ser implantados para executar uma série de tarefas para fornecer serviços sem contato, garantindo que os humanos possam praticar o distanciamento social (SEIDITA et al., 2021). Além de desinfetar ambientes internos (MANTELLI et al., 2022), os RSPs também têm o potencial de apoiar o setor de hotelaria (ROSETE et al., 2020) e fornecer medicamentos e alimentos (LEE et al., 2009; YANG et al., 2020). O uso de RSs em aplicações logísticas é relevante durante esses cenários incomuns. Algumas organizações nacionais dos Estados Unidos identificaram a logística como uma das grandes áreas onde a robótica pode fazer a diferença durante os surtos (SEIDITA et al., 2021).

Em muitos exemplos de aplicações listados acima, é provável que os RSs precisam realizar algumas tarefas de busca. Exemplos simples seriam RSDs buscando e coletando objetos para idosos com restrições de mobilidade e RSPs entregando pacotes em um local específico em um ambiente desconhecido. Semelhante aos humanos no contexto de tarefas de busca de objetos, os RSs também não devem confiar na suposição de que o objeto (ou regiões de interesse) que estão procurando já estão dentro de seu campo de visão (CdV) (SJö; AYDEMIR; JENSFELT, 2012). Portanto, eles precisam encontrar o objeto alvo em ambientes de grande escala com base principalmente em seus sensores visuais, conhecidos como problema de busca por objetos (BPO) (AYDEMIR et al., 2013). No entanto, como um RS encontra o objeto alvo que não está inicialmente dentro de seu CdV? Uma forma de resolver esse problema é fazer com que o RS execute um BPO de força bruta, no qual ele visita todo o ambiente seguindo uma rota de busca pré-definida. Mesmo que essa estratégia pareça uma solução direta, ela não resolve o problema com eficiência (RASOULI et al., 2020). O RS eventualmente o encontrará enquanto o objeto alvo estiver dentro do ambiente. No entanto, o processo de busca pode ser demorado de-

vido às longas distâncias percorridas pelo robô. Outra solução mais eficiente é considerar uma estratégia de busca que incorpore informações tanto do ambiente quanto do objeto alvo, para melhorar o desempenho da busca. Por exemplo, essas informações podem ser a forma da sala para o ambiente (por exemplo, reconhecer uma cozinha e procurar um prato) (AYDEMIR et al., 2011a) e a cor ou categoria/classe do objeto alvo (RASOULI et al., 2020). A estratégia de busca é uma das partes mais críticas de uma abordagem de BPO, pois impacta diretamente na eficiência de um sistema de BPO (AYDEMIR et al., 2013). Portanto, deve ser robusto e eficaz independentemente do ambiente em que o RS está realizando a busca.

A comunidade de pesquisa propôs trabalhos valiosos relacionados ao problema de BPO (EKVALL; KRAGIC; JENSFELT, 2007; SJÖÖ et al., 2009; SJÖÖ; AYDEMIR; JENSFELT, 2012; AYDEMIR et al., 2013; RASOULI et al., 2020). O problema é provado ser NP-Completo (TSOTSOS, 1992; YE; TSOTSOS, 2001), o que significa que a solução de busca ótima pode ser calculada por aproximação (SJÖÖ; AYDEMIR; JENSFELT, 2012), minimizando o custo de busca tanto quanto possível. No caso de RS executando tarefas de BPO, tal aproximação pode ser calculada com o auxílio de fortes pistas fornecidas pela semântica tanto do ambiente quanto de outros objetos no entorno do RS (SJÖÖ; AYDEMIR; JENSFELT, 2012). A semântica pode ser considerada como a informação de alto nível inferida (ou “percebida”) do ambiente, incluindo mas não limitado a nomes e categorias de diferentes objetos, salas e locais (VASUDEVAN et al., 2007; SJÖÖ; AYDEMIR; JENSFELT, 2012; LIU et al., 2016). Da mesma forma, os mapas semânticos codificam não apenas a descrição geométrica e topológica do ambiente, mas também sua interpretação semântica, fornecendo uma maneira amigável para os robôs se comunicarem com os humanos (LIU et al., 2016). Então, quando o RS processa suas leituras de sensores para inferir mais conhecimento sobre seu entorno, ele aumenta o nível de abstração do ambiente ao longo do tempo (BARBER et al., 2018). O uso de informações semânticas e mapas em aplicações robóticas aumenta a autonomia e robustez do robô, além de facilitar algumas tarefas desafiadoras (CESAR et al., 2016).

A.1 Objetivo

Os objetivos desta tese são os seguintes

- *explorar a organização do ambiente e dos objetos para inferir pistas de busca*

semântica para resolver o problema de BPO: pretendemos demonstrar que a informação semântica organizacional pode ajudar no problema de BPO, fornecendo uma maneira de estimar quais regiões são mais propensas a conter o objeto de destino do que outros;

- *inferir informações semânticas organizacionais de números reconhecidos de placas de porta*: argumentamos que o arranjo dos números de placas de porta segue um certo padrão (e regras). Entender como os números estão dispostos permite estimar o quão promissor é um determinado corredor e, então, decidir se a busca deve continuar na região atual ou não. Propusemos um sistema semântico de BPO baseado em números para testar esse argumento;
- *inferir informações semânticas organizacionais a partir de mudanças de objetos semi-dinâmicos*: defendemos que os objetos são movidos principalmente pelos humanos de acordo com suas rotinas e hábitos diários. Isso significa que há grandes chances de que os objetos sejam movidos em um padrão específico ao longo de um período de tempo, sugerindo uma repetição de tempos em tempos. Estimar tal padrão a partir da interação humano-objeto pode ser útil para o problema de SO, para estimar quais locais um objeto alvo pode estar em uma determinada hora do dia. Propusemos um sistema semântico de longo prazo de BPO para verificar essa ideia;
- *realizar uma busca de grosseira para fina de BPO em ambientes internos*: pretendemos propor um sistema de BPO maior que possibilite a um RS encontrar uma sala alvo e interagir com o objeto dentro dela. Ao combinar os sistemas de BPO propostos, acabamos com um sistema de BPO grosseiro para fino, que primeiro procura a sala por seu número e depois procura um objeto de destino.

Afirmamos que, em geral, nossa sociedade não é organizada aleatoriamente, e existem vários padrões e regras que seguimos todos os dias. Não é surpresa que os humanos possam melhorar sua eficiência ao realizar tarefas diárias, como BPO, se o ambiente for meramente organizado logicamente. Assim, eles podem economizar energia e tempo durante essas tarefas. Por exemplo, a maioria das cidades tem suas próprias regras para numerar os imóveis, embora não exista uma regra universal para isso. Os habitantes podem estudá-lo para entender o padrão de numeração local. Assim, eles podem estimar onde está um determinado edifício desconhecido na cidade, mesmo que nunca tenham estado lá. Ao contrário, quando não há regras escritas para especificar a organização do ambiente, os humanos podem entendê-las apenas observando o ambiente por um tempo.

Inspirados no comportamento humano em tarefas de BPO, estamos interessados em fazer com que os RSs aproveitem essa organização de ambiente disponível para melhorar seu desempenho na tarefa de BPO.

Consideramos que as informações semânticas podem ser inferidas do ambiente e podem ser usadas para auxiliar os RSs em tarefas de busca. Essas informações semânticas são úteis para sistemas de BPO porque podem ser usadas como dicas de pesquisa de alto nível. Então, com um sistema de BPO baseado em semântica, o RS não precisaria pesquisar todo o ambiente para encontrar o objeto de destino. O processo de raciocínio humano depende de vários tipos de dicas de pesquisa de alto nível durante as pesquisas, como rótulos e sinais ou o reconhecimento de que outras pessoas podem interagir com o ambiente. Em nossa vida diária, vemos sinais, símbolos e rótulos para avaliar qual direção devemos seguir para encontrar uma sala específica em um ambiente desconhecido. Outro exemplo seria alguém que primeiro verifica se o carro da família está em casa para depois procurar a chave do carro. Em particular, nos concentramos nas informações semânticas organizacionais inferidas da organização do ambiente, como os rótulos e sinais no primeiro exemplo ou o reconhecimento de que outras pessoas podem mover objetos.

A.2 Contribuições desta Tese

Esta tese apresenta resultados (MANTELLI et al., 2021; MANTELLI et al., 2022) mostrando que informações semânticas inferidas da organização do ambiente podem ajudar RSs no problema de BPO. Especialmente, mostramos que o uso de informações semânticas organizacionais como pistas na estratégia de busca de sistemas de BPO pode fazer com que o RS economize recursos por não visitar todo o ambiente. Além disso, mostramos que o uso adequado das informações semânticas pode melhorar a percepção dos RSs para realizar tarefas de alto nível, aproximando-os dos humanos. Também apresentamos uma discussão sobre como nossas contribuições podem ser adaptadas e implantadas no Jaci (MANTELLI et al., 2022). É um robô autônomo higienizador lançado recentemente que visa auxiliar no combate ao COVID-19 e infecções hospitalares devido a contaminações por bactérias e fungos.

A primeira contribuição desta tese é um sistema semântico que realiza a BPO referente à organização da sala (MANTELLI et al., 2021). Tem como objetivo encontrar uma sala específica em um ambiente desconhecido com base na organização das etiquetas das portas. Embora os humanos dependam fortemente de textos, caracteres e símbolos

para realizar várias tarefas, o uso de números de rótulos de texto como fonte de dados não é muito popular na robótica. Neste trabalho, argumentamos que os números detectados a partir de rótulos de texto têm um grande potencial para fornecer pistas de busca e são frequentemente encontrados em ambientes artificiais. Essa ideia surgiu do comportamento humano ao procurar o escritório de alguém em um prédio desconhecido. Mais especificamente, estamos interessados em investigar como os caracteres das etiquetas das portas dos corredores podem ser usados para estimar se um corredor é promissor para encontrar o escritório alvo. A estratégia de busca baseia-se nos padrões de rótulos de portas em cenários internos e raciocina sobre eles para estimar qual corredor é mais promissor para atingir o objetivo.

Outra contribuição é um sistema semântico de longo prazo que busca um objeto alvo em ambientes dinâmicos desconhecidos (MANTELLI et al., 2022). Assume-se que alguns objetos dentro do ambiente nem sempre são estáticos, ou seja, que a organização do ambiente muda ao longo do tempo e está associada às atividades das pessoas. Dessa forma, seu objetivo é incorporar a rotina e os hábitos de uma pessoa na estratégia de busca e, em seguida, fazer as estimativas de busca. Este trabalho teve como objetivo modelar as informações semânticas de como os objetos são organizados ao longo do tempo dentro de um ambiente. Em seguida, ele usa essas informações para evitar que os RSs procurem o objeto alvo em regiões não promissoras. Essa ideia surgiu da observação de como os objetos são colocados ao longo do tempo e que cada pessoa tem suas próprias singularidades em termos de colocação de objetos.

Por fim, finalizamos as contribuições desta tese com uma discussão sobre a implantação de nossos sistemas de BPO no RS Jaci (MANTELLI et al., 2022). Este robô foi construído com o objetivo de ajudar no combate ao COVID-19 e contaminação hospitalar em geral. Está equipado com um conjunto de luzes UV e desinfecta autonomamente os ambientes internos. No entanto, apresenta algumas limitações que podem ser superadas pelos nossos sistemas. Apresentamos nossas ideias de como essa implantação poderia ser feita, juntamente com o possível aumento de eficiência que nossos sistemas podem proporcionar à Jaci.

A.3 Organização

A organização desta tese é o seguinte. Primeiramente, no Capítulo 2, apresentamos a base teórica desta tese, apresentando alguns dos principais problemas da robótica

móvel, juntamente com as abordagens mais populares que tratam de cada problema. Apresentamos também os conceitos gerais do problema de BPO, que é a base desta tese. Por fim, revisamos outras técnicas que são usadas ao longo deste documento. Nos Capítulos 4 e 5, apresentamos os nossos dois sistemas de BPO, BPO semântico baseado em números e BPO semântico de longo prazo, além dos resultados experimentais para ambos os sistemas. No Capítulo 6, apresentamos uma discussão sobre como nossos sistemas de BPO podem ser implantados em um RS real chamado Jaci (MANTELLI et al., 2022). Além disso, também discutimos os benefícios que nossos sistemas proporcionam ao RS em uma aplicação de desinfecção do ambiente.