



TESIS DOCTORAL

Percepción Cognitiva Planificada para Robots Autónomos

Marco Antonio Gutiérrez Giraldo

Tecnología de Computadores y de las Comunicaciones

2017



TESIS DOCTORAL

Percepción Cognitiva Planificada para Robots Autónomos

Marco Antonio Gutiérrez Giraldo

Tecnología de Computadores y de las Comunicaciones

Conformidad de los Directores:

Fdo: Pablo Bustos García de Castro

Fdo: Luis J. Manso Fernández-Argüelles

2017



Planning-based Cognitive Perception for Autonomous Robots

Marco Antonio Gutiérrez Giraldo

Cáceres, 2017

University of Extremadura

Doctoral Dissertation

This work is licensed under license:

[Creative Commons Attribution-Noncommercial-No Derivative Works 3.0.](https://creativecommons.org/licenses/by-nc-nd/3.0/)



Abstract:

The growing presence of robotics in our lives is an undeniable fact backed up by the current prevalence of industrial automation solutions. The next step in this progression will probably be the outbreak of autonomous service robots able to take decisions by themselves. For mobile autonomous robots to become ubiquitous, perception is one of the fundamental skills to improve. Perception is crucial to accomplish intelligent tasks in dynamic environments. For any autonomous system to act intelligently, it must acquire a representation of the environment useful for planning and control, and perception plays a key role in the accurate creation of this kind of representation. On the other hand, planning has a direct effect in the perception outcome through decisions such as viewpoint positioning or sensor parameter tuning. Therefore, since strategies taken by the planners directly affect perception results and vice versa, these two disciplines are meant to understand each other.

Because building fully autonomous robots entails a very wide range of hard and time-consuming problems, they are usually isolated for a faster and more convenient research outcome. Planning and perception are two of these problems that have long been considered isolated. However, the closer we get to develop complete robotic solutions capable of executing intelligent tasks, the more we need these research fields to cooperate in order to obtain more efficient behaviors. Perception plays a key role in the accurate creation of the world internal representation, while planning is of utmost importance for optimal sensor data acquisition and valuable perception results. Therefore, a better integration of these two robotic disciplines is necessary to achieve the successful execution of high level robotic tasks.

The development of a planning-based cognitive perception framework can help robots generate and execute perception strategies to handle high-level uncertainties more efficiently. Existing perception strategies aim for sensor data selection, action parameters determination, selection of techniques to apply, conflict solving and decision making according to the environment and internal knowledge of the robot. Most of these perception processes that are used in the system have a cognitive connotation. This means that the system makes use of internal information of the robot in combination with the sensed data when a search is performed in order to enhance the results. This is called informed search, as it is an aided search that not only uses the data obtained by the sensors in the specific moment of the search but also previous information. Consequently, in this work, when the robot has to search for an object, it takes advantage of the extra cognitive information available. The robot combines information from its own internal knowledge, acquired along its lifetime, with the information obtained from its sensors. The fact that prior information or internal knowledge about objects is considered in every search helps reducing search times, improving the overall results.

The main contribution of the thesis is the development of a system for object informed search using different planned perception processes. The different steps are carefully developed and tested to form a complete system able to make a robot deliver objects in a large household environment. Several techniques were developed for each of these steps to boost the performance of the robot in a complete search and deliver task. The different steps of the cognitive perception are planned accordingly to the goal of the task, the cognitive knowledge in the robot and the environment sensed data. An internal world model in the robot helps maintaining an internal state of the surroundings, while some extra information regarding perceptions might

be stored in the form of deep learning features. Planning is made using this internal model but taking into account the information from the embeddings and other extra sources. This helps maintaining a flexible set of stages that guide the robot through the high level task. It makes the robot able to react to unexpected situations and to finally deliver the requested item. The final goal of this system is to make the robot able to fulfill requests from a human asking it to bring specific objects in a large real environment.

Using the deliberative cognitive architecture CORTEX [Bustos et al., 2016], through its *Active Grammar-based Modeling* (AGM) [Manso et al., 2015], a planning-based cognitive perception framework for autonomous robots has been developed. Several stages of the robot delivery task have been researched, producing a final high level solution tested in real scenarios. Object detection, recognition and location was enhanced through different developments. Object modeling and manipulator motion planning was studied in order to provide a means to reach and grasp the object. Finally a planning platform for all these perceptive steps was developed, enabling the delivery robot to accomplish high level tasks.

Experiments and results are provided for all of the different stages developed along this dissertation. The published papers along with their experiments and results provide support for the scientific impact of the work developed along these years. The experiments and the holistic solution implemented in a delivery robot in a real apartment environment demonstrate the usefulness of the solutions presented here.

Resumen:

La creciente presencia de robots en nuestras vidas es un hecho innegable respaldado por el predominio de las soluciones autómatas industriales. El siguiente paso en esta progresión será probablemente la creación de robots autónomos de servicio capaces de tomar decisiones por sí mismos. Para que los robots móviles autónomos llegaran a extenderse, la percepción es una de las habilidades fundamentales a mejorar. La percepción es crucial para lograr tareas inteligentes en entornos dinámicos. Para que cualquier sistema actúe de manera inteligente, debe adquirir una representación del entorno que sea útil para la planificación y el control. La percepción juega un papel esencial en la creación precisa de esta representación. Por otro lado, la planificación tiene un efecto directo en el resultado de la percepción a través de decisiones como el posicionamiento de puntos de vista o ajustes de parámetros del sensor. Por lo tanto, como las estrategias tomadas por el planificador afectan directamente los resultados de percepción y vice versa, estas dos disciplinas están destinadas a entenderse.

Por el hecho de que la construcción de robots autónomos completos implica un gran número de problemas difíciles, normalmente son aislados para mayor rapidez y una investigación más conveniente. La planificación y la percepción son dos de estos problemas que durante mucho tiempo se han considerado aisladamente. Sin embargo, cuanto más cerca nos encontramos de desarrollar soluciones robóticas completas capaces de ejecutar tareas inteligentes más necesitamos que estos campos de investigación cooperen con el fin de obtener comportamientos más eficientes. La percepción juega un papel importante en la creación de la representación interna del mundo, mientras que la planificación es de alta importancia para una adquisición de datos óptima por parte del sensor y para obtener buenos resultados de percepción. Dicho esto, una mejor integración de estas dos disciplinas robóticas es de gran importancia cuando se intenta conseguir la ejecución satisfactoria de tareas robóticas de alto nivel.

El desarrollo de un framework para la planificación de la percepción cognitiva puede ayudar a los robots a generar y ejecutar estrategias de percepción para manejar incertidumbres de alto nivel de manera más eficiente. Estas estrategias de percepción tratan de seleccionar datos del sensor, determinar parámetros de las acciones, seleccionar técnicas a aplicar, resolver conflictos y tomar decisiones teniendo en cuenta el entorno y el conocimiento interno del robot. La mayoría de estos procesos de percepción que se usan en sistema tienen una connotación cognitiva. Esto significa que el sistema usa la información interna del robot en combinación con los datos de los sensores cuando se ejecuta una búsqueda con el fin de mejorar los resultados. Esto se llama búsqueda informada, ya que es “ayudada” pues no sólo usa los datos del sensor en el momento específico de la acción. Consecuentemente, en este trabajo, cuando el robot tiene que realizar una cierta búsqueda, se aprovecha de la información cognitiva. El robot combina información de su propio conocimiento interno, adquirida durante su vida, con la información de los sensores. El hecho de que se considere información anterior o conocimientos internos sobre los objetos durante las búsquedas, ayuda a mejorar dichas tareas en tiempo y resultado.

La principal contribución de la tesis es el desarrollo de un sistema para la búsqueda informada de objetos usando diferentes procesos de percepción. Los diferentes pasos han sido cuidadosamente desarrollados y probados con el fin de formar un sistema completo que permita al robot encontrar y entregar objetos en grandes entornos domésticos. Varias técnicas han sido desarrolladas para cada uno de los pasos para mejorar la actuación del robot en una tarea completa de búsqueda de objetos. Los diferentes pasos de la percepción cognitiva son

planificados de acuerdo al objetivo de la tarea, el conocimiento cognitivo del robot y los datos de los sensores. Un modelo interno del mundo en el robot ayuda a mantener un estado interno del entorno, mientras que alguna información extra sobre percepciones puede ser almacenada en forma de características de “deep learning”. La planificación utiliza este modelo interno pero considera la información de las características y otras fuentes. Esto ayuda a mantener un conjunto de pasos flexibles que guían al robot a través de la tarea de alto nivel. Esto permite que el robot sea capaz de reaccionar a situaciones inesperadas para finalmente entregar el objeto requerido. El objetivo final del sistema es proveer al robot con capacidades para cumplir peticiones de un humano que le pida que le traiga objetos específicos en grandes entornos reales.

Usando la arquitectura cognitiva deliberativa CORTEX [Bustos et al., 2016], a través de su *Gramáticas Activas basadas en Modelado* (AGM) [Manso et al., 2015], se ha desarrollado un framework de planificación de la percepción cognitiva. Se han investigado diferentes pasos de la tarea de búsqueda y entrega del robot, produciendo una solución de alto nivel probada en escenarios reales. La detección, reconocimiento y localización de objetos ha sido mejorada a través de distintos desarrollos. El modelado de objetos y la planificación del movimiento de manipuladores ha sido estudiada con el fin de aportar métodos para alcanzar y agarrar el objeto. Finalmente una plataforma de planificación para todos estos pasos perceptivos ha sido desarrollada, permitiendo al robot reparto cumplir sus tareas de alto nivel.

Se aportan experimentos y resultados para los diferentes pasos desarrollados durante esta tesis. Los artículos publicados así como sus experimentos y resultados respaldan el impacto científico del trabajo desarrollado en estos años. Las últimas pruebas y experimentos con la solución holística implementada en un robot de reparto en un apartamento real prueban la utilidad de las soluciones aquí presentadas.

Contents

1	Introduction	1
1.1	Why planning-based cognitive perception?	1
1.2	The case study: A delivery robot	3
1.3	Motivation	4
1.3.1	Environment Sensing and Cognitive Modeling of Rooms	5
1.3.2	Semantic Relations for Scene and Object Discovery	7
1.3.3	Modeling and Planning for Grasping	12
1.3.4	Informed Search for Planning Perception	14
1.4	List of Publications	16
I	Publications	19
2	A Cost-efficient 3D Sensing System for Autonomous Mobile Robots	21
	Introduction	23
	Previous Works	23
	System Design	24
	Hardware	24
	Data Processing	25
	Experimental Results	27
	Mapping	27
	Novelty Detection and 3D Shape Retrieval based on Gaussian Mixture Models and Superquadrics	28
	Conclusions and Future Work	30
	Acknowledgements	30
	References	30
3	An Incremental Hybrid Approach to Indoor Modeling	31
	Introduction	33
	Overview of the Approach	33
	Rooms and Doors Modeling	34
	Room Detection	35
	Door Detection	36
	Incremental Modeling of the Environment	36
	Experimental Results	37
	Conclusions and Future Work	39
	Acknowledgement	39

References	39
4 A Passive Learning Sensor Architecture for Multimodal Image Labeling: An Application for Social Robots	41
Introduction	43
State-of-the-Art	45
Passive Learning Sensor Architecture	46
Cognitive Attention	47
Cognitive Subtraction	48
CNN Classification Step	49
Semantic Processing	50
Experiments	52
Tests on Image Buffering	53
Cognitive Attention Tests	54
Tests with Generic ImageNet Training	55
Tests with Networks with Fine-Tuned Training Datasets	56
Conclusions and Future Work	58
Abbreviations	59
References	59
5 Semantic Exp. of Auto-Gen. Scene Desc. to Solve Robotic Tasks	63
Introduction	64
System Design	65
Multimodal Encoder-Decoder Pipeline	65
Word Semantics Relationships	65
Word Matching System	66
Experiments	66
Conclusions and Future Work	68
Acknowledgement	68
References	68
6 Exploiting Symmetries and Extrusions for Grasping Household Objects	71
Introduction	72
Related Work	73
Generating Compact Object Representations from Single RGB-D Images	73
Superquadric Representation	73
Object Completion from Extrusions	74
Using Compact Object Representations for Grasp Planning	75
Experimental Results	76
Accuracy of Fit	76
Robot Grasping Experiments	77
Discussion and Conclusion	77
Acknowledgements	78
References	78
7 SPAM for a Manipulator by BNM in Unknown Environments	79
Introduction	80

CONTENTS

Best Next Move Planner	81
Simulation Results	83
Experiments	84
Conclusions and Future Work	85
References	85
8 Perceptive Parallel Processes. Coordinating Geometry and Texture	87
Introduction	88
Related Works	89
The Perception System	89
Texture Aware Perceptive Process	90
Multimodal Neural Model	90
Syntactic Frequency Distribution Parser	90
Geometry Aware Perceptive Process	91
Looking for Tables	91
Object Recognition and Pose Estimation	91
Experiments	91
System Setup	91
Results on the Experiments	92
Conclusions and Future Works	93
References	93
9 Integrating Planning Perception and Action for Informed Object Search	95
Introduction	96
Related Works	97
Integration of an Object Oracle in a Robotics Architecture	98
Domain: Node Types and Actions	99
Agents	99
Design of the Oracle Agent	100
Cue Acquisition	101
Semantic Container Vector Representation	103
Querying Oracle for Object Location	103
Experimental Results	104
Conclusions	105
References	
II Conclusions and Future Works	97
10 Review and contributions	99
11 Future works	101
A Robotic Software Contributions	103
A.1 The Needs of Specific Software for Robotics	103
A.2 Component Oriented Programming	105
A.2.1 Main Characteristics	105

CONTENTS

A.2.2	Advantages and disadvantages	106
A.2.3	Why Component Oriented Programming for Robots?	107
A.3	Model Driven Engineering for Robotics	109
A.4	Robotic Frameworks	112
A.4.1	RoboComp	114
A.4.1.1	Component Model	115
A.4.1.2	Libraries, Tools and Files	115
A.5	Robot Perception Software	116
A.5.1	Point Cloud Library	117
A.5.2	OpenDetection	118
B	Publications not covered in this thesis	121

Chapter 1

Introduction

“ Chew, if only you could see what I’ve seen with your eyes! ”

— Roy Batty, Blade Runner

1.1 Why planning-based cognitive perception?

Science fiction has promised us robots capable of cleaning houses or performing normal day to day activities. The industrial success of automation and teleoperated robots confirms that current robots are physically capable of performing almost any motion. These robots operate in industrial environments, performing high precision tasks and lifting heavy loads. However, when dealing with day to day tasks in unknown environments and when being outside of carefully controlled setups, even the most sophisticated industrial robot would be unable to carry a simple object to a specific place. For them to achieve full autonomy, a key element is still missing: a perceptive-cognitive structure to help them perform tasks autonomously. A structure like this would allow robot maintaining internal representations of the environment and concepts related to it that would help them reasoning and making decisions when working toward a certain goal.

Nonetheless, building fully autonomous robots entails overcoming a wide range of hard and unsolved problems. In order to deal with the challenge and come up with results without having to solve all the issues, researches have split up the problem into smaller pieces and focused their research on each of them separately. Therefore, most research groups often focus only on one or two of these fields and do not consider the rest. While isolating the problems is a good way to achieve successful research results, when the goal is to have a fully autonomous robot working in a real environment, it comes to a point that some of these skills have to be integrated. This increases the research complexity, but, on the other hand, it can help achieve new successful results, as more information can be taken into account than when using isolated approaches. In fact, some researchers are starting to embrace this novel multimodal procedure, leading to the recent trend in more holistic robotics solutions.

While in certain areas of robotics such as control or motion planning there is still work to do, it is undeniable that one of the core pieces that we are currently missing is a good perception system. Perception is a key component when it comes to performing almost any day to day task. Moving robots in unknown environments requires a good perception to avoid hitting obstacles.

Any kind of process involving detection, classification and/or location calls for an autonomous system with good visual skills. Even after the searched object is detected, perception is also needed to properly reaching it and to proceed with the final grasp. Therefore, since most of the main tasks an autonomous robot could perform depend to a large extent on a proper perception, solving this issue becomes of utmost importance for the progress of the research in robotics.

Sensors play a key role when it comes to the performance of perceptive processes. Different aspects like quality, quantity or type of data sensed highly impact the robot's understanding of the environment. In the last few years, the affordability and performance of PrimeSense devices have made easy for anyone to have access to 3D data. The previous solutions for sensing depth information were either costly, self-made [Gutierrez et al., 2011] or dependent on the scene information, such as stereo vision. However, still most of the current perception solutions consider texture information separately from geometric information. Prove of this is the existence of separated solutions and libraries such as OpenCV [Bradski, 2000] and the PointCloudLibrary [Rusu and Cousins, 2011]. The OpenDetection library [Sarkar and Gutierrez, 2016] is an effort to unify geometrical and visual perception with the specific needs of robotics. Detection and recognition tasks may require texture information, as we may need to differentiate between objects with the same shape. At the same time, shape is important to detect objects with little or no texture, and for manipulation tasks. But as intended through this research, the combined use of geometry and texture information can lead perception systems to more robust outputs.

Deep Learning [Goodfellow et al., 2016] comprehends a set of Machine Learning techniques that allow the representation of complex concepts from simpler representations, *i.e.*, a trained Deep Neural Network (DNN) can encapsulate small representations of different features of images that, when used in combination, build up to more complex concepts. Recent advances in hardware, along with the huge amount of data available nowadays in the cloud, have boosted the results provided by these networks. Works in this area, like [Szegedy et al., 2015, Krizhevsky et al., 2012, Simonyan and Zisserman, 2014] or [Girshick et al., 2016], achieve astonishing results in image classification when trained and tested with standard datasets such as the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al., 2015] or the PASCAL VOC [Everingham et al., 2010]. However, when these trained networks face real world environments, with their inherent handicaps such as unfavorable light conditions, low resolution or messy and cluttered scenes, the outcome is often not good enough to be used in a production system. Consequently, testing and improving these solutions in robots, dealing with real scenarios, becomes a key asset when trying to strengthen and enhance perception systems results for real applications.

Since perception is somehow present in almost any service the robot performs, it has to adapt to every state and requirements from the system in order to advance toward higher level goals. For this reason, creating a planned perception that takes into account all the different sources of information would enhance the overall perception task performance. For example, a planned perception would allow the robot to take proper decisions regarding the usage of geometry or textured information according to the stage of the plan and the needs of the robot itself at that moment. Also the feedback coming from the perception level needs to have a direct impact into the planning in order to be able to adapt to the environment in non-trivial situations. For instance, the robot might need to acquire a better viewpoint of an object due to poor light conditions or to obtain a complete visual model of it. This work represents an effort to solve this problem through the production of a flexible plan while exploiting cognitive information. To achieve this it tries to plan and adapt the vision system, on every stage of the plan, using the

cognitive and environmental data.

1.2 The case study: A delivery robot

From the moment a human asks a robot for an object, until the point when it delivers the requested item, lots of different processes take part in a “robotic mind”. Even though it might look like an easy task for a human being, for a robot, it is a long and complex series of actions that have to be executed under the uncertainty of a real environment. These actions are executed, within the robot, as numerous processes that have to robustly interact either with the environment (through sensors or actuators) or with each other. Therefore, in order to make these processes and resulting actions able to cope with the uncertainty of the environment, while working toward a certain goal, the whole cognitive perception process must be carefully planned.

In this dissertation, planning-based cognitive perception is presented through the specific case of a human asking a robot to bring a certain object located in a large household environment. Location of the object is unknown, although some information might be contained in the internal world model. The robot uses this information combined with the data obtained from sensors in order to plan and execute the different perceptual steps, taking into account the outcome of the previous stage and the state of the environment in order to execute the next one.

When robots search objects in wide environments, as a first step, in order to narrow the search space, it has first to decide where to look for a certain object. A basic approach would be to go around exploring the unknown frontiers of the search space [Yamauchi, 1997]. However, when looking at humans visual performance, it seems that long term memory representations have a strong influence on it [Woodman and Chun, 2006]. In this work this same principle is applied to robotics in the form of informed search. Informed search helps the robot take advantage of its previous experience and knowledge when taking early decisions on where to begin a certain search.

Subsequently, once the robot reaches possible locations, a set of actions should be triggered in order to look for the object. In order to detect the different elements, segmentation of the scene must be properly achieved. After segmentation is done, a proper classification process that labels each of the parts is needed to provide the robot with a detailed knowledge of the objects in its surroundings. Different approaches to the detection and localization problems were developed and enhanced in order to provide the robot with these capabilities for the whole search process.

Finally, after the robot locates the object within its surroundings, it must grasp it to later perform its delivery. For this to happen not only proper location of the object is needed, but some extra geometrical information is also required in order to plan a proper grasping of the object. Object modeling techniques were developed to provide this stage with the necessary information for a successful grasp. Likewise, an arm motion planning and grasping should be performed. A Simultaneous Planning and Mapping (SPAM) algorithm was developed for this matter. It helps the robotic arm to reach the object through a proper path avoiding collisions. Only then, a robot can reach the object, grasp it and deliver it to the human requesting the object.

Shelly (figure 1.1) is the name of the delivery robot used in the experiments of this dissertation. It is the fourth generation of a manipulator robot, completely build and designed by the researchers at RoboLab (the robotics lab at the University of Extremadura, Spain). As being



Figure 1.1: The Shelly robot looking for a requested object in the kitchen of the apartment.

fully designed locally there is less dependencies on third parties for hardware maintenance. It is also good for fast and custom updates, keeping the design up to the state-of-the-art and always matching the researchers needs. The main drawback is that there is a higher workload in the research process however, on the other hand, more is learned in the process.

It is equipped with an omnidirectional base that allows the Shelly to move in any direction at any point. This helps avoiding unnecessary turns and reducing the accumulated error in odometry measurements. On top of the base lies a 2D LRF that is used for mapping and localization purposes as well as obstacle avoidance. The robot is equipped with 4 Intel NUC PCs for onboard processing, interconnected through a switch and accessible through a wifi connection. Additionally, these computers can be managed through a touch screen located at the front of the robot. This screen can also be used for human-robot interaction purposes. It has a robotic arm with four degrees of freedom and a grip for object grasping. A motorized head along with several sensors are also installed in the robot. Current available sensors are a camera, primesense device and a kinect v2, although they can easily be exchanged for new ones when needed.

1.3 Motivation

The ensemble of research articles produced along the development of this dissertation creates a well defined pipeline that empowers autonomous systems to produce object deliveries in apartment like environments. We present here and tie together these works with the aims of understanding them as a whole research production. The analysis of the research is performed through the use case of the delivery robot.

Initial works try to cover the first needs of the delivery robot, sensing the environment and modeling rooms for SPAM, presented in works outlined in subsection 1.3.1. Once the robot is capable of sensing and modeling, word semantics relations are exploited to boost object scene discovery results in the research works presented in subsection 1.3.2. Once the object to deliver has been found, a proper modeling of the object along with a robotic arm path planning is

needed in order to be able to grasp the object. This is analyzed in subsection 1.3.3. Finally optimization and management of the whole delivery robot is done by means of planning-based cognitive perception in the papers described in subsection 1.3.4.

1.3.1 Environment Sensing and Cognitive Modeling of Rooms

The first and most crucial task to perform when a robot needs to know its surroundings is to sense the environment and obtain data for further processing. Nowadays different sensing systems exist to obtain 2D and 3D data, being the most popular the PrimeSense sensor family due to its great cost-efficiency balance. These sensors offer decent 3D colored information of the environment at a rate of 30 frames per second (fps). However until the mass production of this solution, cameras were the main cost-efficient sensor. Back then, due to sensor access most perception solutions were focused on 2D image processing. 3D information was built using 2D images mostly through stereo vision techniques with its own drawbacks (*i.e.*, the need of texture or calibration). Although several methods provide different means to estimate depth from RGB images [Ayache and Lustman, 1991] [Meguro et al., 2007] [Moravec, 1996] [Scharstein et al., 2001], they still have a strong dependencies on light variations and texture. Other solutions like 3D LRFs (Laser Range Finders) provide much more accurate data with less environmental dependencies at higher cost. Before the boost of the PrimeSense devices, 3D sensor alternatives to cameras was high and out of budget for most robotics labs. In these conditions an effort to build “**A Cost-Efficient 3D Sensing System for Autonomous Mobile Robots**” (see chapter 2) was made to conceive an affordable alternative to the existent costly 3D sensors.

The sensor developed in this work consists of a complete 3D sensing system solution for autonomous mobile robots. A regular 2D LRF sensor is used and moved by a step motor in order to obtain 3D points from a full 360° scan. Additionally, a regular camera is mounted on top and used to obtain texture data from the environment. This allows the system to provide the RGB information attached to each of the 3D points produced by the LRF scan. Finally, an embedded system is configured to manage the different components of the system. The software to control the system is built on top of RoboComp, the component framework used for robotics development through this dissertation (see section A.4.1). A camera component handler, along with another one for the LRF, run in the embedded system, managing and sending the data over the network to a more powerful desktop for high level processing. The embedded system converts coordinates from Polar System to Cartesian system and adds color through an extrinsic 3D-LRF Camera Calibration.

In order to test the system, a mapping of the environment was performed using the Chen-Medonig (point-to-plane) framework for the Iterative Closest Point (ICP) algorithm [Chen and Medioni, 1992]. 24 different scans, moving the robot around the lab with a rotation $\beta > 2$ rads, were made. Data results matched accurately the environment shape and measurements. However, it was discovered that when rotation was increased over 0.2 rads different point clouds were not smoothly connected. This was attributed to high rotational differences among point clouds that are hard to handle by this ICP algorithm. Furthermore, using our system the Gaussian Mixture Model (GMM) for novelty detection [Drews et al., 2010] was deployed and tested. In this development a multi-scale burden was initially performed in order to reduce computation load. Afterwards, the Earth’s Mover Distance [Rubner et al., 1998] was used over the GMM in order to detect differences among scans. This enabled the system with the capability of detecting new objects added in the scenes. Figure 1.2 shows an execution of

this algorithm in a room scanned with the 3D sensor. Figure 1.2a corresponds to the scan of an empty room along with the detected Gaussians. The second scan of the room, this time including the novelty, along with the associated Gaussians, is shown in figure 1.2a. Finally, once the novelty was detected, a superquadric was shaped in order to fit the point cloud and provide the robot with an idea of the shape of the new object.

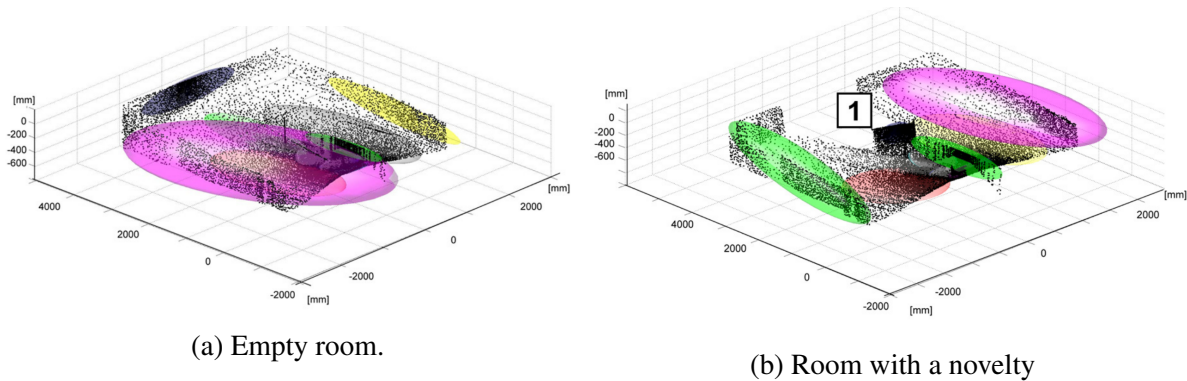


Figure 1.2: Downsampled point cloud scans and associated Gaussians.

Once the low level data acquisition is functional, the delivery robot needs to locate within the environment in order to move around it. For the robot to be able to locate itself, a proper mapping of the environment is needed. Using our new developed sensing system, **An Incremental Hybrid Approach to Indoor Modeling** (see chapter 3) was developed. This work enables the delivery robot to model rooms along with the doors that connects them into a novel hybrid cognitive representation. With this modeling information the robot is able to locate itself and produce paths in large indoors environments when looking for objects.

In the representation used there, the model of the environment is stored as an undirected graph whose vertices represent the different rooms and edges the doors connecting them (see Figure 1.3). This cognitive representation allows the robot to easily reason over large environments with several rooms and doors (*i.e.*, the robot can obtain a minimum path connecting two rooms). On top of that, the representation can also be extended to contain more complex world structures such as different floors in a building. This topological representation avoids the need of a metric map of the environment since it only has to maintain the parametric description of each room and doors. Developed geometric restrictions and loop closings help correct possible errors in the final model derived from noise in the environment.

In this work the problem of room modeling was reduced to a rectangle detection problem as rooms are assumed to be rectangular with walls perpendicular to the floor. Therefore, a new rectangle detection technique was developed based on a search in the parameter space through a 3D variation of the Hough Transform [Rosenfeld, 1969, Joo et al., 2010]. In the detection process, only the points that belong to the contour of a rectangle are considered. The pose of the robot relative to the room where it is located is computed according to the perceived rectangle regions. For more details and a formal definition of the method please refer to chapter 3.

Similarly to the apartment like environment, in which final tests of our delivery robot are performed, the indoor modeling algorithm is tested in a two rooms environment connected through a door (see figure 1.4a). Starting from the middle of one of the rooms, the robot is made to move around in order to incrementally model the first room. Once the it is modeled, the robot proceeds through the door and models the second room. Finally after the loop closing

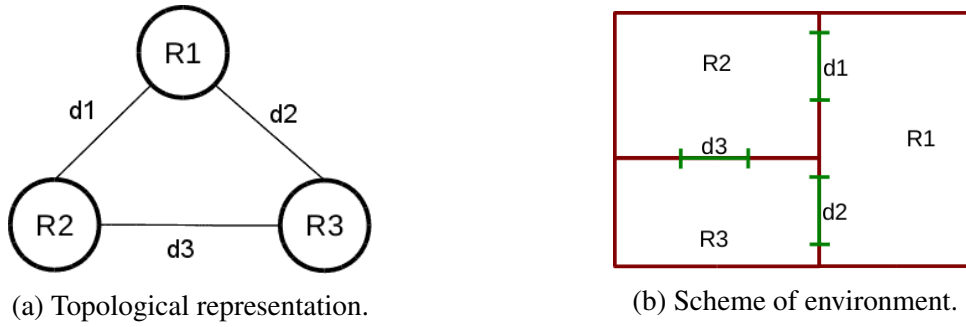


Figure 1.3: Topological representation of an environment composed by three intercommunicated rooms.

and application of restrictions (*i.e.*, door matching), both rooms were properly modeled (see figure 1.4). Each square of the figure represents an area of $0,27 \times 0,27m^2$. The real sizes of the two rooms were $3,19 \times 3,78m^2$ (room 1) and $4,20 \times 3,78m^2$ (room 2) and the sizes obtained by the modeling process were $2,97 \times 3,78m^2$ (room 1) and $4,05 \times 3,78m^2$ (room 2). Although the accuracy of each room model is limited by the sampling step of the Hough space, the resulting error is in the permissible range for the needs of the delivery robot. If needed, more accuracy can be obtained by reducing the sampling step size of the Hough space (at the expense of a higher CPU load).

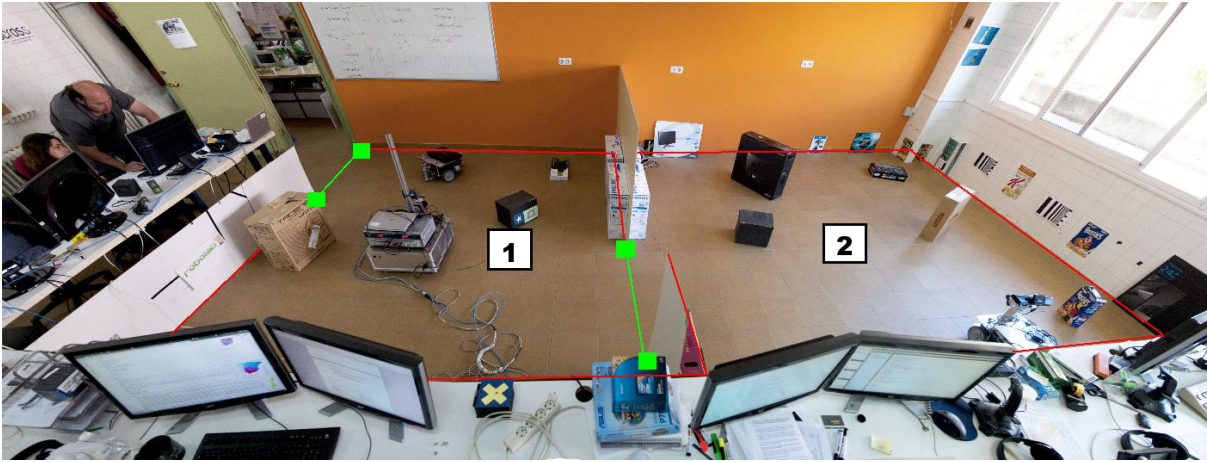
The delivery robot was enabled with sensing, mapping and location capabilities. Using the hardware and software developed it should, at this point, be able to retrieve information of the environment and produce an accurate cognitive map to locate itself and move around.

1.3.2 Semantic Relations for Scene and Object Discovery

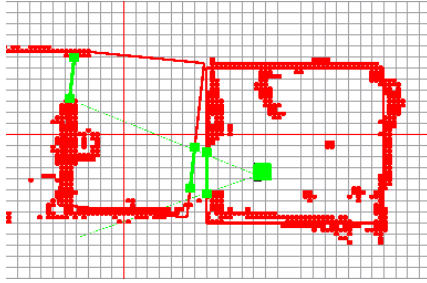
Once the robot is able to start moving around and look for the target object, it should take decisions on where to go. Choosing where to start and how to approach different locations is not a trivial task. Making good decisions at this stage can enormously reduce the time taken to find and reach the object. Exploiting word semantics can help boost the significance of objects labels and infer new information that helps the robot understand scenes. In fact, the use of semantics is a promising approach for scene labeling as confirmed by several works in the area such as [Romero-González et al., 2017] or [Rangel et al., 2016].

In an effort to select the best potential locations within rooms to find a specific object the research presented in “**A Passive Learning Sensor Architecture for Multimodal Image Labeling: an Application for Social Robot**” (see chapter 4) was conducted. In this work, a four stages architecture is developed to exploit informed search in order to optimize the initial selection of containers to speed up the full object search process. The Passive Learning Sensor Architecture (PLSA) is designed to take advantage of the multimodal information obtained when combining RGB-D sensor data with trained semantic language models. The main contribution of this work lies in speeding up the search process using data acquired in real environments and from distant locations. This means, the objects sensed data, acquired by the robot, will have a low resolution and poor light.

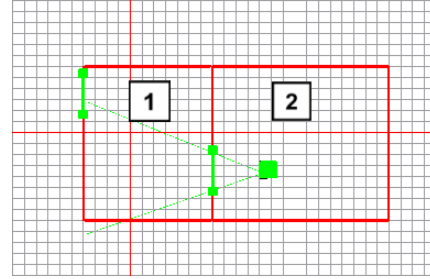
Figure 1.5 shows the four stages of PLSA. The first stage, cognitive attention, detects object containers in images discarding those without them. From the selected images, the region of the container is segmented and provided to the second step. For this segmentation, the cognitive



(a) Frontal view of the two rooms with connecting door.



(b) Modeling of the two rooms before restrictions are applied.



(c) Final resulting model of the two rooms.

Figure 1.4: Modeling of two contiguous rooms with a connecting door used in the modeling experiments

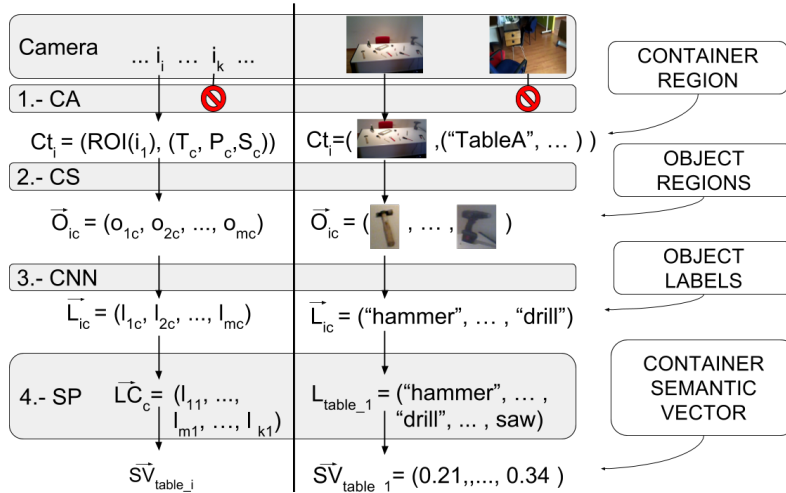


Figure 1.5: The four main steps of our passive learning architecture. The left-hand side of the vertical black line describes the output of each step in a mathematical notation, while the right-hand side shows it visually. Explanations of the outputs are given on the outer right descriptions. The forbidden sign means the image will be discarded.

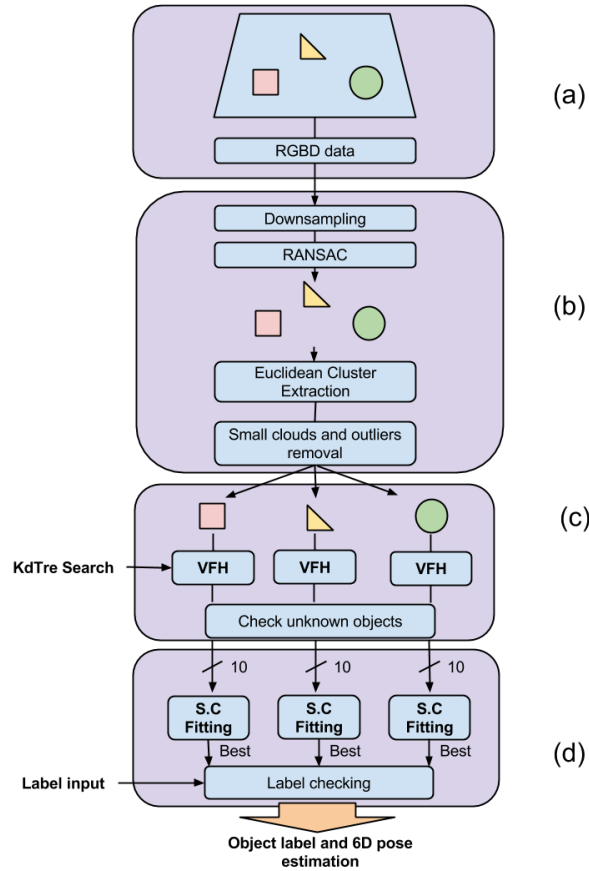


Figure 1.6: Tabletop segmentation and object recognition pipeline used by the texture aware process.

information regarding the robot and possible object containers location is used. Afterwards, the cognitive subtraction step performs a segmentation process that outputs the regions corresponding to each object lying in the container. This step is specific to each container, in particular, for tables, the pipeline shown in figure 1.6 is used. Once objects are segmented, they are labeled using a very deep Convolutional Neural Network (CNN) based on deep residual learning [He et al., 2016] trained with ImageNet [Russakovsky et al., 2015]. These labels are then processed by the Semantic Processing step. This step uses word semantics, in a similar way as the work in chapter 5, to compute an average semantic vector that represent each container. All the vector representations of the labels produced by the previous step for a certain container are processed to produce that container’s average semantic vector. Finally these average vectors are compared with the representation of the label of the object to find to determine the optimum order in which tables should be visited.

Similarly to the work explained in chapter 3, the experiments are also performed with the delivery robot in a two room apartment-like real environment. Five containers (tables) are disposed with objects on them, grouped according to their use, as shown in figure 1.7. The tests preformed tried to prove the usefulness of each of the stages present in the architecture. Initial tests showed that a buffer of about 20 images was enough for PLSA to acquire optimum performance. PLSA outperformed state-of-the-art segmentation algorithms combined with CNN



Figure 1.7: Setup of the experiment: At the top the real setup, bottom-left is the cognitive model of the setup, bottom-right are images of the real tables and objects.

architectures, trained with generic datasets and specially tuned for our experiment. Finally, the semantic step was tested, showing improvements in the results respect to a non-semantic step architecture. However, an interesting discovery was that this step does not improve that much the results of the CNN architectures with retrained models. Further details are available in chapter 4.

Once possible object locations are selected and approached by the robot, the object recognition stage should be triggered. In order to help the delivery robot with the detection of objects a **“Semantic Expansion of Auto-Generated Scene Descriptions to Solve Robotic Tasks”** was developed (see chapter 5). This work analyzes and enhances auto-generated image descriptions in order to detect semantic relations between a query and a scene. The system takes an object description as input and produces a set of matching scene images using word semantic expansion techniques in auto-generated image captions.

The first part of the system is in charge of generating realistic image descriptions. A multimodal encoder-decoder pipeline is in charge of generating these descriptions (see figure 1.8). The encoder learns from a joint image-sentence dataset. Sentences are encoded using a long short-term memory (LSTM) recurrent neural network, as described in [Hochreiter and Schmidhuber, 1997]. Image features are extracted from the top layer of a deep convolutional network trained with the Imagenet dataset for the classification task [Krizhevsky et al., 2012] and projected to the multimodal embedding space. A pairwise ranking loss is minimized in order to learn to rank images and their descriptions. For decoding, a Structure-Content Neural Language Model (SC-NLM) [Kiros et al., 2014] is used. It is a multiplicative neural language model where the attribute vector is an additive function of the embeddings. These embeddings are conditioned on the embedding vector for the description computed by the LSTM.

After a set of scene descriptions are generated, semantic relations are used to help find matchings between the descriptions and the system query. An improved version of the skip-gram model [Mikolov et al., 2013] was used for this matter. This version models word representations that help predict the surrounding words in a corpus. As more than often related

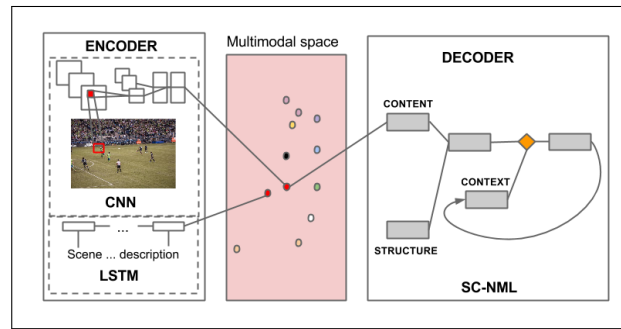


Figure 1.8: Multimodal encoder-decoder pipeline.

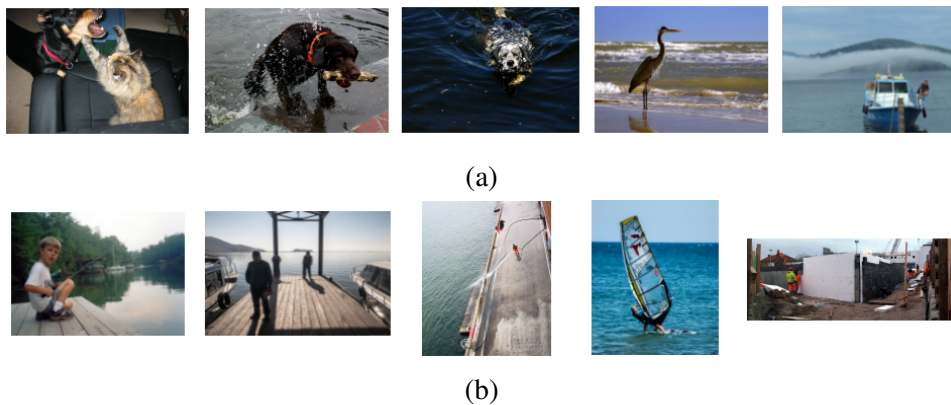


Figure 1.9: Top results of the query *look for a pet in a river*. a) These are the results using the word semantics relations. None of the generated captions specifically contained either the word *pet* or *river*. b) These are the results for the direct matching experiment. Only five of all the generated captions contained the word *river*.

words are found next to each other in corpus, when training the model with large datasets, it tends to capture semantic relationships among words. Descriptions are syntactically analyzed using NLTK and key parts are selected while the rest are discarded (*i.e.*, connectors or articles). An average of the model representations of these key words is computed for each image. Then this representations are compared using the cosine distance with the representation of the query and images are ranked according to their results. Objects can be searched using descriptions that might contain negative parts *e.g.*, “not red”). These negative parts are identified in the syntactic analysis process and the representation is subtracted, instead of added, to the final average vector value. This process moves away the description average vector from the vector representation of that word, making it more unlikely to get good results with semantically related words.

Experiments for this work compare qualitative results of the ranking of images and a regular direct matching among words in the query and words in image descriptions. Figure 1.9 shows a test for the query *look for a pet in a river*. Using the developed system, images containing a pet and water are usually ranked high, even though neither the word “pet” nor the word “river” were present in the generated captions of these images. However, for the direct match approach only the word “river” could be found in some captions, leaving no option to even consider any

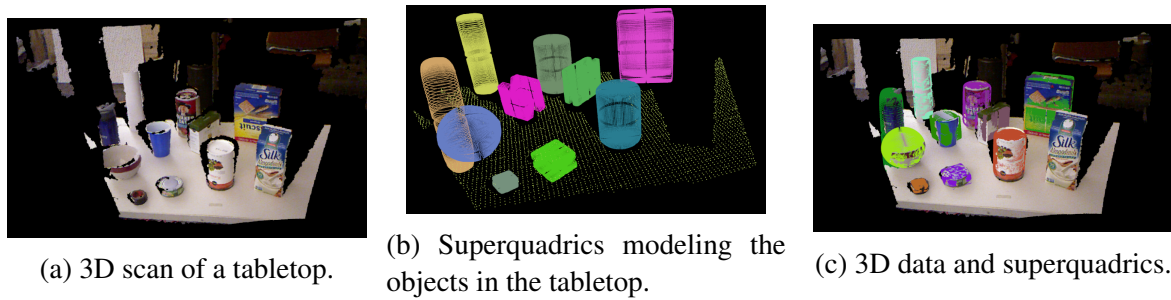


Figure 1.10: Superquadrics modeling of objects on a tabletop.

image containing pets (as the word was not present at all on any image caption of the dataset). Further examples can be found in the experiments sections of chapter 5.

1.3.3 Modeling and Planning for Grasping

Once the robot recognizes and locates the object to deliver, the first steps toward a grasping maneuver take into action. The first work in this direction is an effort on **“Exploiting symmetries and extrusions for grasping household objects”** (see chapter 6). This work introduces algorithms for completing partial point clouds through the analysis of the symmetry and extrusions that the shape of different objects present. The robot is empowered with the ability to predict the 3D shape of an object from a single view to prepare for a final grasping step. Two shape generation techniques are developed in this work, one using superquadrics and the other one exploiting linear extrusions.

The superquadric representation used here is similar to the one in chapter 2, although this time is enhanced with point cloud completion through symmetries detection techniques. This is done because superquadric fitting techniques such as [Duncan et al., 2013] have their performance decreased when applied over 3D data corresponding to one view of an object. Although hardcoding limits in axes dimensions can help overcome this issue it is still not practical for real environments where the shapes of different objects can have high variations. Therefore, similarly to [Bohg et al., 2011], an additional pre-processing step was added in an effort to provide a more complete point cloud to the superquadric minimizations process. This point cloud completion step finds an optimal symmetry plane perpendicular to the table where the object is lying in order to mirror the points. Figure 1.10 shows a tabletop of objects modeled using this technique.

For the extrusions based shape completion, a three-step approach was used to calculate the extrusion axis. Points from the initial point cloud are used to generate a first hypotheses of the extrusion axis. Afterwards, a line is fit into the set of hypotheses using linear least-squares along with a RANSAC algorithm [Fischler and Bolles, 1981]. Finally, an optimization on this initial estimate is performed using the dynamic hill climbing algorithm [Yuret and De La Maza, 1993]. Once optimized, points are rotationally extruded around the axis to generate the complete point cloud. Using the Poisson surface reconstruction [Kazhdan et al., 2006] on the completed point cloud a mesh fitting the object is obtained. Extracted extrusion information is later used to reduce the complexity of the grasp planning.

Experiments on this work measured the accuracy of the fittings and performed full grasp planning tests with a humanoid robot. Symmetric shapes were transformed into partial clouds

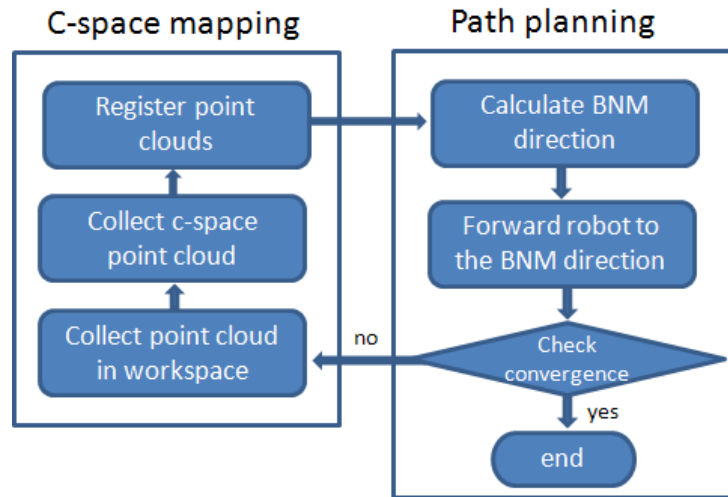


Figure 1.11: Simultaneous Planning and Mapping cycle

with noise to simulate a single 3D view. An error distance of 2mm was produced when completing the shape using extrusions while a higher error was found for the superquadrics shape fitting (slightly larger than original shape). Finally, a set of grasping experiments were performed on four objects. For more details and comments on the experiments please refer to chapter 6.

Once the object shape is ready and the possible grasps are generated the arm needs a path to reach the target. In order to help this process a “**Simultaneous Planning and Mapping (SPAM) for a Manipulator by Best next Move in Unknown Environments**” (chapter 7) was developed. This work aims to provide the manipulator robot with SPAM capabilities in order to move in unknown environments. A skin type setup based on 3D depth camera sensors that completely encompass the manipulator is used to obtain 3D point clouds. These clouds are then used to create a c-space map in which the Best Next Move (BNM) algorithm is applied to direct the motion of the manipulator and avoid obstacles.

Similarly to Best Next View [Torabi and Gupta, 2010], in BNM the local motion in each step is designed to reveal the maximum environmental map possible. BNM uses the strategy in [Temerinac et al., 2007] for point cloud registration as it calls for rapid identification and collection of 3D shapes. Map construction and goal convergence are considered as separated goals to be addressed in parallel. For map construction, global motion is steered toward maximum environment mapping. In fact, the mapping in BNM has a tendency to move toward the direction by which the point clouds spread in space, thus easing a faster mapping of c-space obstacles. For further details on the implementation of BNM please refer to chapter 7. For a visual overview, figure 1.11 shows the full proposed SPAM cycle with the two parallel processes.

Several simulation experiments were conducted to test the proposed SPAM algorithm. BNM was tested against sensor-based RTT [Um et al., 2013]. After 30 runs, the total time, the number of point cloud generated and the mapping efficiency was measured. BNM took half the time, generated almost the same amount of point clouds but doubled in mapping efficiency the results from sensor-based RTT. A final test with a real manipulator and two IPA sensors [Um et al., 2011] was performed proving maximum environment map generation and goal reaching.

1.3.4 Informed Search for Planning Perception

Once the different steps for an object search were developed the next logical development was to integrate them into a full delivery robot solution and test the utility of the whole system. A first work in this direction was done through **Perceptive Parallel Processes Coordinating Geometry and Texture** (see chapter 8). This work presents a full solution for a robot looking for objects in large household environments. Two processes run in parallel, separately analyzing the geometry and texture of the 3D and 2D information obtained by the robot.

In order to find the best potential places to start the search process, the system starts performing a syntactical analysis on the request given to the delivery robot. The Neural Language Toolkit (NLTK) [Bird et al., 2009] is used to analyze the search query and obtain the target object. Immediately after, the texture-aware perceptive process analyzes a database of images that correspond to generic versions of the rooms available in the environment. This means that if one of the rooms available for the delivery robot is a kitchen, a set of kitchen images will be analyzed to decide whether this room should be visited first. This allows the robot to pre-evaluate the potential rooms to visit without wasting time trying to reach them. In this initial step, the texture process generates automatic descriptions of these generic images and evaluates the appearance of the searched object in them. The description generation is made through a multimodal neural model following the structure in [Kiros et al., 2014]. Afterwards, these descriptions are analyzed and a frequency histogram that captures the appearance of the different nouns is generated. Finally, this is compared to the object label extracted from the query to provide a certain score for the room. The same process is repeated for each room, scoring each of them and producing a final ordered list of places to visit for the robot.

Once the order of rooms to visit is determined, the delivery robot starts checking them one by one. During this stage the two parallel processes, geometric and texture, get started. The first one, has a focus on texture analysis and aims for a rapid wide scene labeling. This helps taking decisions on what places should be further inspected or if, on the contrary, the robot should give up in that area and continue with the next spot. Even though the analysis of the scene is similar to the one used for the room selection step. The difference lies in that the texture process uses the camera images that the robot captures instead of generic ones.

On the other side, the geometry-aware perceptive process performs a more detailed evaluation of the scene. A pipeline similar to the one in chapter 8 and shown in figure 1.6 is used. This process has a table detection step that exploits geometry restrictions in order to detect them as planes at a certain height and parallel to the floor. Afterwards, a tabletop segmentation mechanism is used to select the object on the table (see Figure 1.6.b). This step is followed by a geometry matching verification using Viewpoint Feature Histograms (VFH) [Rusu et al., 2010] to recognize the shape of the different objects and estimate their location. Finally, a label matching step selects the proper object from the table if present. If the object is not found in the processed table the robot would mark the table as visited and proceed with the processing scenes and tables.

For the experiments on this work an environment larger than previous ones was used. This time, a full apartment with seven different locations was explored by the delivery robot (see Figure 1.12). Five executions of five different object searches were performed, making a total of 25 tests performed. The first step, the order of places to visit, was first evaluated, resulting in a total success, always selecting the room containing the object as a first target. Afterwards, the full recognition and location of the object was evaluated, with a success rate of 92% of the 25

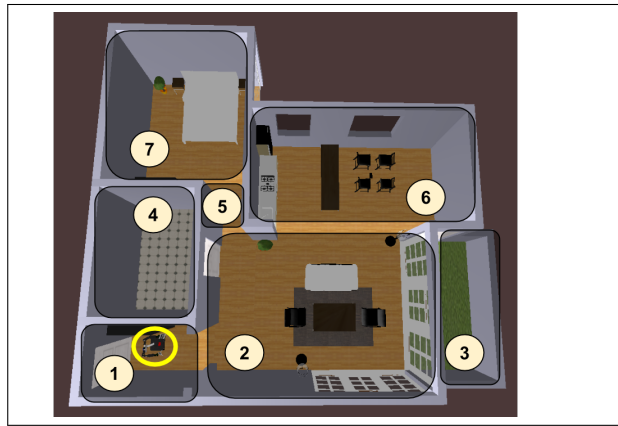


Figure 1.12: An overview of the household environment used in the experiments. The rooms are labeled as follows: 1.- Entrance, 2.- Living room, 3.- Patio, 4.- Bathroom 5.- Hallway, 6.- Kitchen, 7.- Bedroom. Circled in yellow is the robot at its starting point.

test cases. More details on these experiments can be found in chapter 8.

Another full solution was the effort to integrate the architecture explained in chapter 4 into a planned perception system in order to produce a holistic delivery robot solution. For this, a work in **“Integrating Planning Perception and Action for Informed Object Search”** (see chapter 9) was developed. This research comprises the planning and execution of all the perception steps needed for the delivery robot to work. It integrates PLSA (see section 1.3.2) into an oracle agent that aids the delivery robot decide what containers to approach. A final verification step, using the same structure as the geometry-aware process from previous work (chapter 8), is also added in order to certify the existence of the object. Once the object is recognized and located, the robot triggers the object grasping step to grab the object for the final delivery.

The deliberative cognitive architecture, CORTEX [Bustos et al., 2016], enables the robot to perform all the steps of the object search process. These steps start by asking the passive learning agent (oracle agent) for object location hypotheses. Later on, the cognitive architecture guides the robot through the search, following with the verification of the object classification and its precise location within the scene to finish grasping the target. CORTEX is built on top of many developments carried out during past years, the most relevant being the *Active Grammar-based Modeling* architecture (AGM) [Manso et al., 2015], the RoboComp framework [Manso et al., 2010, Gutiérrez et al., 2013] or the Deep State Representation concept [Marfil et al., 2014, Manso et al., 2016].

Planning is made using the world model of the robot and taking into account the information from the deep learning features along with other external sources. This helps maintain a flexible set of stages that will guide the robot through the high level task. This flexibility makes the robot able to react to unexpected situations and creating a robust set of actions to finally deliver the requested item.

The higher-level modules called *agents* share information through the internal world model. This world model helps maintain an internal state of the surroundings, while some extra perceived information might be stored in the form of deep learning features. This model combines symbolic and metric information as necessary [Manso et al., 2016] and is managed and mon-

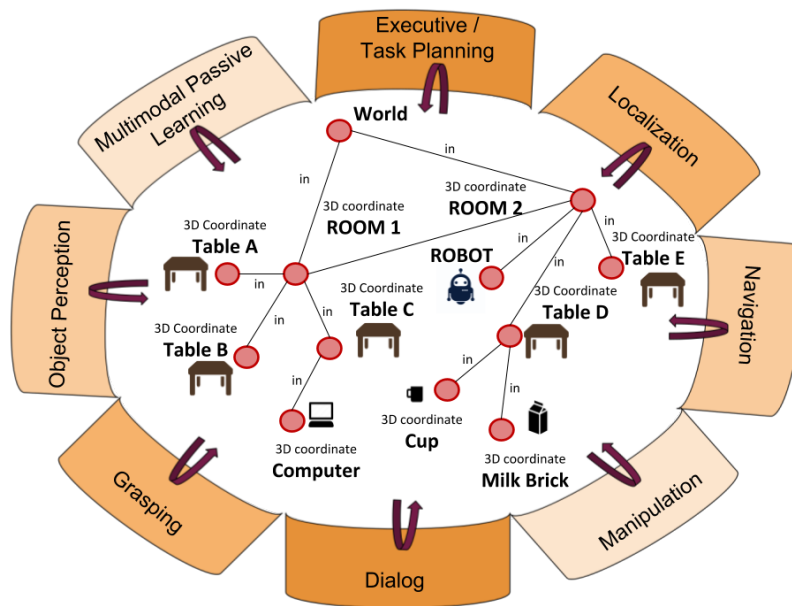


Figure 1.13: The CORTEX architecture along with its agents accessing and contributing to the maintenance of a shared representation of the environment and of the robot itself. Planning and Executive are in charge of the high-level task planning activities, although all agents might have domain specific deliberative capabilities.

itored by an *executive* module in charge of providing agents with a sequence of *actions* that, if executed correctly, would take the robot to a state where its goal has been achieved (object found). As a result the *executive* generates a plan that correspond to the different steps of the cognitive perception that considers the goal of the task, the cognitive knowledge in the robot and the environment sensed data. Specifics on how the architecture performs the planing of the cognitive perception process can be found in chapter 9.

For the experiments, the same setup as in the work from chapter 4, shown in figure 1.7, was used. The delivery robot was placed in random positions within the two rooms and asked to find five objects from five different locations. The proposed method was tested against a random and traveling salesman algorithm. The probabilities of finding the object in the first place were of 0.36, 0.35 and 0.9 for the random, traveling salesman and oracle-based solutions, respectively. The average time take for the robot to reach the proper container was 82.68, 61.0 and 26.13 seconds for the random traveling salesman and oracle-based solutions. Time taken for table inspection was ignored as it is not part of the proposed architecture and it is constant for all the tested algorithms. Even though the random and traveling salesman policies have similar success rates, the latter is able to approach the right table because it optimizes the order in which tables are inspected. More details on these results are available in chapter 9.

1.4 List of Publications

Environments Sensing And Rooms Cognitive Modeling:

- “A Cost-efficient 3D Sensing System for Autonomous Mobile Robots”. **Marco A. Gutiérrez**, E. Martinena, A Sánchez, Rosario G. Rodríguez, P. Núñez. In Proc. of Workshop of Physical Agents, WAF 2011, September 2011. Albacete, Spain.
- “An Incremental Hybrid Approach to Indoor Modeling”. **Marco A. Gutiérrez**, P. Bachiller, L. J. Manso, P. European Conference on Mobile Robots, ECMR 2011. September 7-9, 2011. Örebro, Sweden.

Semantic Relations for Object Detection:

- “Perceptive Parallel Processes Coordinating Geometry and Texture”. **Marco A. Gutiérrez**, Rafael E. Banchs and Luis F. D’Haro. Proceedings of the Workshop on Multimodal and Semantics for Robotics Systems, IROS 2015. pp. 30-35. 2 October, 2015. Hamburg, Germany.
- “Semantic Expansion of Auto-Generated Scene Descriptions to Solve Robotic Tasks”. **Marco A. Gutiérrez**, Rafael E. Banchs. International Journal of mechanical Engineering and Robotics Research. vol. 5, no 2, p. 109. April, 2016.

Modeling and Planning for Grasping:

- “Exploiting symmetries and extrusions for grasping household objects”. Ana Huamán Quispe, Benoît Milville, **Marco A Gutiérrez**, Can Erdogan, Mike Stilman, Henrik Christensen, Heni Ben Amor. IEEE International Conference on Robotics and Automation (ICRA) 2015. pp. 3702-3708. 26-30 May 2015. Seattle, Washington.
- “Simultaneous Planning and Mapping (SPAM) for a Manipulator by Best Next Move in Unknown Environments”. Dugan Um, **Marco A. Gutiérrez**, Pablo Bustos and Sungchul Kang. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, November 3-8, 2013. Tokyo, Japan.

Informed Search for Planning Perception:

- “A Passive Learning Sensor Architecture for Multimodal Image Labeling: An Application for Social Robots.” **Marco A. Gutiérrez**, Luis J. Manso, Harit Pandya and Pedro Núñez Sensors 17, no. 2: 353.
- “Integrating Planning Perception and Action for Informed Object Search.” Luis J. Manso, **Marco A. Gutierrez**, Pablo Bustos and Pilar Bachiller. Cognitive Processing (To Appear), 2017.

Part I
Publications

Chapter 2

A Cost-efficient 3D Sensing System for Autonomous Mobile Robots

A Cost-Efficient 3D Sensing System for Autonomous Mobile Robots

Marco A. Gutiérrez, E. Martinena, A. Sánchez, Rosario G. Rodríguez, P. Núñez.

Abstract— This paper describes a mechanism for building an inexpensive and, at the same time, accurate system for 3D scanning on Autonomous Mobile Robots. Our system allows us to obtain 3D points from the robot environment along with its associated color. This data can be later processed using different techniques in order to obtain information from surrounding objects useful for tasks such as navigation or localization. Information is obtained at a rate of 50 ms per line of scan (700 points per line). In order to use the sensor as part of an active perception system, resolution is made to be directly dependent on the scanning speed and robots are able to adjust the related parameters accordingly to their needs. Our approach uses a regular commercial 2D Laser Range Finder (LRF), a step motor and a camera, all this controlled by an embedded circuit which makes the system apt for being built in any regular Autonomous Mobile Robot. Finally, to test our system, two different real applications have been used. First a 3D Map reconstruction is done using several point clouds matched by the ICP algorithm and our odometry. Then, we make a novelty detection and 3D shape retrieval using the Gaussian Mixture Model and Superquadrics.

Index Terms—Autonomous Mobile Robots, 3D Shape retrieval, Mapping, Laser Range Finder, RGB-D

I. INTRODUCTION

Sensing and processing the unknown environment is crucial for Autonomous Mobile Robots to obtain information from their surroundings. Most of the actions a mobile robot could achieve, such as mapping, localization, exploration or navigation, have strong dependences on the information obtained from their environment. Hence, the task of properly acquiring and processing this information has become a critical need in the mobile robotics field.

In order to obtain this information Autonomous Mobile Robots, can use different sensing systems to achieve data. Cameras are one of the most common used interfaces to obtain environment data. They have been widely studied and, therefore several algorithms are available to achieve world modeling. For this reason, not only direct information is now obtained from cameras, other, such as depth, can be estimated following several methods [1], [2], [3], [4], however most of these solutions have a necessity for texture information. Therefore, these algorithms have a strong dependency on light variations and wouldn't work properly in indoors environments specially those with surfaces not uniformly colored. LRF (Laser Range Finder) solutions provide a more accurate choice in terms of precision and environmental dependency. However, they lack from texture information when it is available.

Marco A. Gutiérrez, E. Martinena, A. Sánchez, Rosario G. Rodríguez and P. Núñez are with University of Extremadura.
E-mail: marcog@unex.es

The lack of good, cheap and fast sensors allowing robots to sense the environment in real-time, in order to allow them to act on the basis of acquired data, is one of the reasons of the gap between prognoses and reality. Several commercial accurate 3D LRF systems are already available in the market. However, most of them have usually a high cost (~ 50.000 USD). This way, we have developed a 3D sensing system for Autonomous Mobile Robots. It consists on a 2D LRF moved by a step motor, a camera for texture information and an embedded system that manages the other three components. Although this managing choice could limit, in some way, the whole system it has been chosen that way because it is intended to be used in Autonomous Mobile Robots and they have a strong needs for small and power-efficient sensing systems. The embedded system is in charge of directly moving the step motor, acquiring information from the LRF and the camera, collecting the 3D data and sending it over the network for its processing, storage or visualization.

This paper is organized as follows. In section II we talk about some previous works in the field. Our 3D sensing system is deeply explained in Section III, its hardware components and how the data is processed. Then we test our system using two different applications in Section IV. First building a 3D map of the robot environment using the Iterative Closest Point (ICP) algorithm and the odometry information, and then detecting changes on 3D maps applying the Gaussian Mixture Model and retrieving shapes from detected objects using superquadrics. Finally, in Section V we present some conclusions and future work directions.

II. PREVIOUS WORKS

Autonomous Mobile robots have a strong need of 3D ranging sensors. Therefore researchers have been increasingly focusing efforts in this field, and, as a consequence, groups have came up with different approaches to obtain 3D points and shapes from the environment.

The relative low cost of cameras have made these systems very common. Therefore, using cameras to obtain 3D information has been a widely spread research effort. In order to obtain depth information from images the number of cameras used have changed along the different studies. The most popular solution is using two cameras (Binocular Stereo), mostly inspired in the way humans obtain depth perception from their environment. Several groups have done intensive works on these systems [2] [3] [4]. But also other number of cameras such as three [1], one (monocular cameras) [9] or even only using unsorted collections of images [10], have been

used. However, these systems are highly dependent on texture information. This makes them to, usually, loose accuracy when facing indoor environments. In the same way, most of the time, these solutions have high computational costs with big power requirements and a small field of view (usually 60°).

In order to solve the lack of texture dependency the use of laser lines projection have taken into account [8]. Even some RGB-D commercial sensors widely popular nowadays such as the Primesense RGB-D sensor [20] make use of infrared projection to reduce this texture need. Although solutions are very popular, compared to LRF performance, they get small fields of view, low depth precision (3cm in 3m scan) and high sensibility to light variations.

Due to the price difference from commercial 2D LRF to 3D LRF, solutions to achieve the whole 3D spectrum with 2D LRF systems have been explored. This has been done either making the 2D LRF scanner rotate itself [12] or rotating a mirror in front of the it [13]. Some of these solutions even include a camera for texture information retrieval [14]. On our system, efforts have been not only focused on the LRF-camera 3D system but also on managing it from an embedded system in order to make it able for running on small and low power consumption Autonomous Mobile Robots.

Finally, some other different 3D retrieval solutions are worth to mention, like using a rotating camera and an angled LRF [15] or even using camera shadows to construct Multiple Virtual planes [11].

III. SYSTEM DESIGN

To develop a complete sensing system that can cover most of the mobile robotic possible needs, a wide range of possibilities must be taken into account. Indoor or outdoor scenes, more or less light or uniformly or not uniformly colored objects are some of the different aspects from the environment a robot can face. Our device is made up in an effort to take into account all these likelihoods, trying to get the best and more accurate info out of each situational environment.

Our design is intended to be simple and low cost ($\sim 3500\text{€}$). And although, some of the hardware could be cheaper (laser is $\sim 3000\text{€}$), it has been chosen that way due to our target of getting the most available information out of the surroundings a robot could find itself, while, at the same time keeping it small and power-efficient to be deployed on Autonomous Mobile Robots.

A. Hardware

Our solution consists of three main hardware parts, showed in Fig. 1. First, a commercial 2D LRF is moved by a step motor (labeled as (1) and (2) on Fig. 1) to be able to obtain 3D points from full 360° scans. Secondly, a camera (labeled (3) on Fig. 1) is used in order to achieve texture feedback from the environment. When this information is made available, we are able to attach color information to the correspondent points obtained from our LRF. And finally, an embedded system (labeled (4) on Fig. 1) that manages the rest of the components in the system. Fig. 1b shows a sketch of the system and the associated reference frames of some of the elements.

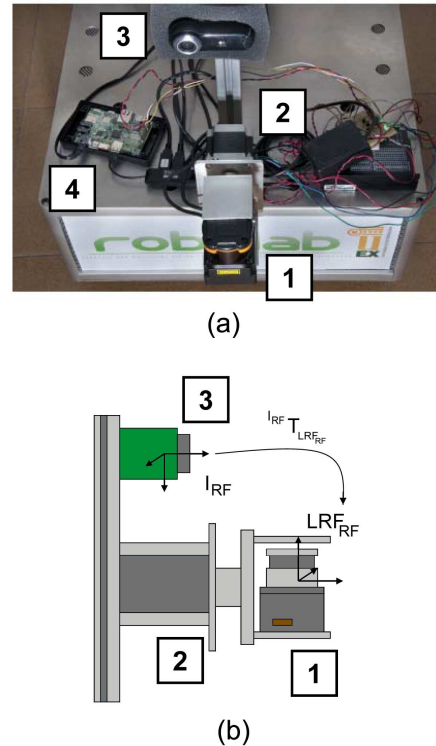


Fig. 1: a) The 3D sensing system mounted on one of our Autonomous Mobile Robots, *Robex*. The three main hardware parts are marked in the Figure (LRF (1), step motor (2), RGB camera (3) and embedded system (4)) b) Sketch of the measurement system and associated reference frames.

We use a Hokuyo LRF with a large scanning range (30 meters and 270°). It has been chosen that way so that we can make scans either indoor or outdoors with appropriated accuracy, although the system is fully compatible with almost any other 2D LRF sensor. The LRF is attached to a step motor to make it scan the full 360° in front of the mobile robot. The step motor has enough torque to move the LRF without using any gear-train and, in consequence avoiding the backlash they usually introduce. It has 200 steps resolution and it is attached to a power driver to obtain a higher one, up to 25000 steps. Between the several ways of moving a laser that exist, we have chosen to do it in the Z axis, with $\alpha=0$ degrees, as shown in Fig. 2. This is the solution that better fits our needs because it leads to high resolution of points in front of the robot, exactly the place it is facing and most possibly to where it is moving [7]. This allows us to focus our density distribution on certain objects, i.e. obstacles or new elements introduced in the environment [18]. The resolution we want to achieve on the scan is directly dependent on the speed of the step motor. Therefore, our system is good to be used for active perception purposes since most of the resolution is kept

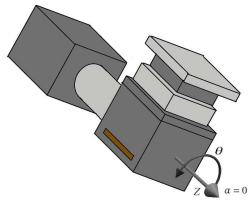


Fig. 2: LRF and step motor rotating scheme.

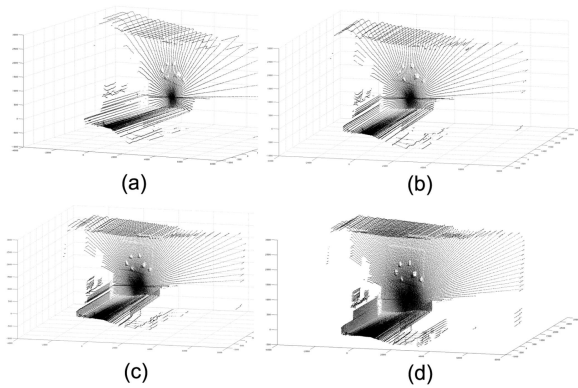


Fig. 3: Scans using different speeds, and therefore achieving different levels of resolution: a) 0.6 rad/s, ~ 25 lines of scan b) 0.4 rad/s, ~ 50 lines of scan c) 0.2 rad/s, ~ 100 lines of scan d) 0.1 rad/s, ~ 200 lines of scan.

in front of the robot and is able to select the proper speed for each scan, changing the resolution in consequence. Fig. 3 shows different resolutions scans from our system, according to selected speed and its approximated scan resolution.

The Camera consists of a regular Web-Cam that provides us with 640x480 images at a rate of 10 frames per second. It is kept statically on top of the LRF and the step motor facing the same space as the scanner in order to match the color pixels from the camera with their correspondent 3D points from the LRF. The camera is USB connected to the embedded system to make more accurate and real-time matching of the captions and the 3D points. Both two sensors have different reference systems, I_{RF} and LRF_{RF} (see sketch at Fig. 1b). Therefore, as mathematically shown in Section III-B, data information from the camera would be calibrated, that is, processed to match the LRF one. This process involves finding the transformation $I_{RF} T_{LRF_{RF}}$ (Fig. 1b).

Finally, all these elements are controlled by our embedded system. It has a *GNU/Linux* distribution and several *Robocomp* [5] components on top of it. It takes responsibility for properly moving the step motor according to given instructions and, therefore, assuring the required resolution. It makes calculations to retrieve the angles and data coming from the LRF. Besides, it captures images from the camera and assigns pixels to the corresponding 3D points. Then, information can be

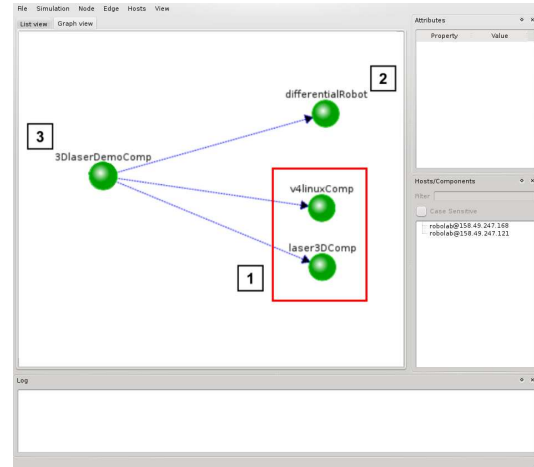


Fig. 4: Screenshot of the *managerComp Robocomp* tool. 1) The 3D laser and the Camera running on the embedded system 2) The component in charge of the odometry running on the robot 3) The process and display component running on a Desktop system.

locally stored for later processing, or sent through the network for live treatment, storage and/or display.

B. Data processing

The software to control our system is built in on top of the robotics framework *Robocomp* [5]. Making use of its component oriented programming and its communication middleware we are able to minimize the CPU load in our embedded system leaving the heavy processing for the powerful desktop systems. A light-weight camera component handler and another for the LRF and the step motor are executed directly in the embedded system (see (1) on Fig. 4), sending the data over the network to other components running on desktop computers that will make further data storage, analysis and/or display (see (3) on Fig. 4). Thanks to the communication middleware the system constitutes a generic component that can be used through its interface:

```

1 interface Laser3D
2 {
3   void moveToAndSweepAtSpeed (float
      minAng, float maxAng, float speed,
      bool once, int numScans, out int
      nScans, out float period);
4   void stop();
5   TLaserData getLaserData();
6   Laser3DConfData getLaserConfData();
7 }

```

Listing 1: Interface of the *Laser3D Robocomp* Component

1) *Coordinate systems conversion*: The 2D LRF returns points in the Polar Coordinate System (Fig. 5a) that is, a distance r and a polar angle or azimuth $\varphi \{(r_i, \varphi_i) | i = 1..n\}$. Then, making use of the information coming from our step

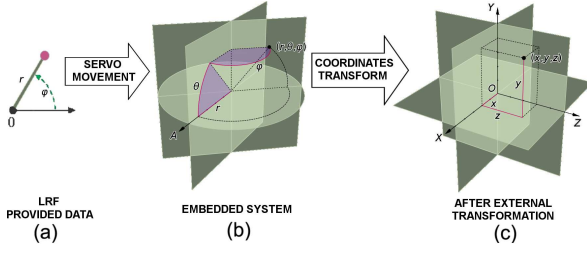


Fig. 5: Transformation from the LRF data to Cartesian Coordinates.

motor we obtain the inclination angle θ of the LRF. This leaves our coordinates expressed in the Spherical System (Fig 5b), having a radial distance r , an azimuth angle φ and an inclination angle θ , $\{(r_i, \varphi_i, \theta_i) | i = 1..n\}$. Then, data is finally converted into the Cartesian System $\{(x_i, y_i, z_i) | i = 1..n\}$ (Fig. 5c), using the following regular systems conversion equation:

$$\begin{cases} x_i = r_i \cdot \sin\theta_i \cos\varphi_i \\ y_i = r_i \cdot \sin\theta_i \sin\varphi_i \\ z_i = r_i \cdot \cos\theta_i \end{cases}, \quad i = 1..n \quad (1)$$

It is desired to keep the embedded system as much free of CPU load as possible to assure a good real-time response. However, since our inclination angle θ is directly dependent on the step motor movements information and we want to preserve the system accuracy, it is unavoidable but to execute the polar system coordinates to the spherical system transformation on our system. Therefore, transforms up to the spherical coordinates are processed on the embedded system, then data is sent over the network and the rest of the processing steps are computed on external and more powerful systems.

2) *Extrinsic 3D-LRF Camera Calibration:* Camera image data is also obtained. It comes in form of RGB matrix that is matched to its correspondent laser points. For this matching Rotation (R) and Transformation (T) of those color points to the laser reference frame must be performed [6]. In order to calibrate our camera-LRF, calculation of proper R and T is needed. This R and T are needed to obtain the target (X^L, Y^L, Z^L) , therefore we can say:

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} = R \begin{bmatrix} X^L \\ Y^L \\ Z^L \end{bmatrix} + T \quad (2)$$

Where (X^c, Y^c, Z^c) are points on the reference system of the camera and (X^L, Y^L, Z^L) the ones from the laser. Also written as:

$$\begin{aligned} X^L &= r_{11}X^c + r_{12}Y^c + r_{13}Z^c + T_x \\ Y^L &= r_{21}X^c + r_{22}Y^c + r_{23}Z^c + T_y \\ Z^L &= r_{31}X^c + r_{32}Y^c + r_{33}Z^c + T_z \end{aligned} \quad (3)$$

We obtain n empirically known matching pair points in our LRF and camera, in the form:

$$((X_i^L, Y_i^L, Z_i^L), (x_i, y_i)), \quad i = 1..n \quad (4)$$

Where every point (x_i, y_i) obtained from the image matches the corresponding (X_i^L, Y_i^L, Z_i^L) obtained from the LRF.

Following [6] we could conclude that using equation 5 we can obtain a solution.

$$Av = 0 \quad (5)$$

Where A is defined in equation 6 with an $n \geq 7$, and being the points not coplanar.

$$A = \begin{pmatrix} x_1 X_1^L & x_1 Y_1^L & x_1 Z_1^L & x_1 & -y_1 X_1^L & -y_1 Y_1^L & -y_1 Z_1^L & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n X_n^L & x_n Y_n^L & x_n Z_n^L & x_n & -y_n X_n^L & -y_n Y_n^L & -y_n Z_n^L & -y_n \end{pmatrix} \quad (6)$$

This solution depends on a parameter or scale factor:

$$V = (v_1, \dots, v_8) \begin{cases} v_1 = r_{21} & v_5 = \lambda r_{11} \\ v_2 = r_{22} & v_6 = \lambda r_{12} \\ v_3 = r_{23} & v_7 = \lambda r_{13} \\ v_4 = T_x & v_8 = \lambda T_x \end{cases} \quad (7)$$

Imposing the rotation matrix orthogonality we could determine the mentioned matrix and two components of the translation vector T_x and T_y . Then, to obtain the last component of the translation vector, T_z , we only have to solve equation 8.

$$B \begin{pmatrix} T_z \\ f_x \end{pmatrix} = b \quad (8)$$

Where:

$$B = \begin{pmatrix} x_1 & (r_{11}X_1^L + r_{12}Y_1^L + r_{13}Z_1^L + T_x) \\ \vdots & \vdots \\ \vdots & \vdots \\ x_n & (r_{11}X_n^L + r_{12}Y_n^L + r_{13}Z_n^L + T_x) \end{pmatrix} \quad (9)$$

and

$$b = \begin{pmatrix} -x_1(r_{31}X_1^L + r_{32}Y_1^L + r_{33}Z_1^L) \\ \vdots \\ -x_n(r_{31}X_n^L + r_{32}Y_n^L + r_{33}Z_n^L) \end{pmatrix} \quad (10)$$

Then we obtain T_z following:

$$\begin{pmatrix} \hat{T}_z \\ \hat{f}_x \end{pmatrix} = (B^t B)^{-1} B^t b \quad (11)$$

This way we have obtained all the components of the correspondent Rotation Matrix R and Translation Vector T which are used to transform the points from one system to the other or, in the same way, assign the texture information to the 3D laser points (see Fig. 1b).

Fig. 6 shows a scan of our lab taken at $0.2\pi \text{ rad/sec}$ of speed (10 seconds per 360° scan). The image is shown at Fig. 6a and data points from the same scan at Fig. 6b. After, as

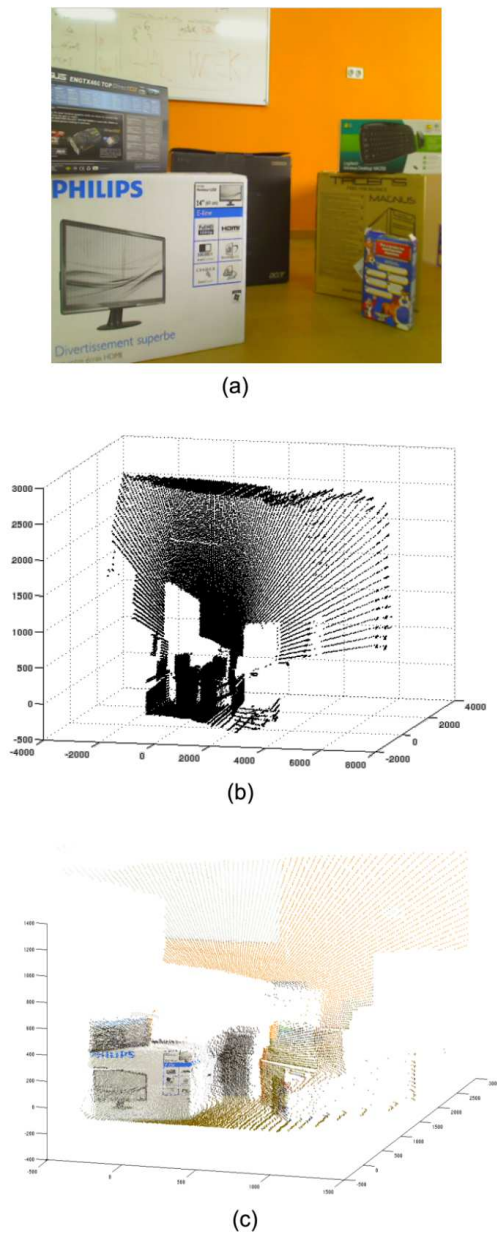


Fig. 6: (a) Camera image from a scan in our lab (b) Points from our 3D LRF (c) Colored data from the scan after data transformation.

explained, properly matching both data sets a result such as the one shown in Fig. 6c is obtained. Notice that colored data is not the whole 3D data from LRF, since the camera covers a much smaller area than the LRF, as explained on Section II.

IV. EXPERIMENTAL RESULTS

We have built our testing system using a Hokuyo UTM-30LX 2D LRF. It has 30 meters and 270° of scanning range. The LRF scans 270° at a rate of 25ms. However, we get the scans at a speed of 50ms due to the use of our, in some way, CPU limited embedded system. The 2D LRF is attached to a CeNeCe 23HB56 step motor to achieve the whole 3D spectrum. The motor has 200 steps and a 11 Kg/cm torque and it is attached to a Leadshine DRP452 driver with 15 different modes of resolution (from 1/2 to 1/125 steps). A Logitech Quickcam Pro 9000 Web Camera is used to obtain texture information from the environment. It gives us 640x480 images at a rate of 10 FPS (Frames Per Second). This all is controlled by our embedded system consisting of an ARM Cortex-A8 OMAP 3530 of 720 Mhz and 4G Mobile Low Power DDR SDRAM @ 200 Mhz running an Ubuntu GNU/Linux System. All of it is properly mounted and assembled into *Robex*, one of our Autonomous Mobile Robots (see Fig. 1).

The software have been developed using the *Robocomp* framework [5], which makes it easy to balance the work load between the embedded system and other more powerful ones. On the embedded system we have deployed two main components, one for managing the step motor and laser system and other for managing the camera. This data is served through the network to another external components running in more powerful desktop machines. Then, as said, the collecting data component either processes, stores or shows data for its further analysis.

We have tested our 3D sensing system using two different algorithms. First we have run a scan matching in an effort to make a map of the environment of the robot. For this algorithm we have tried obtaining different scans at different points of the room and then matching them using either the ICP scan matching algorithm [16] or odometry.

The second experiment consists of a novelty detection on a 3D map and a subsequent shape retrieval of detected novelty using superquadrics. We have used an algorithm that simplifies the data using a multi-scale sampling technique in order to reduce the computation time of detecting changes in the environment. Then a method based on the Earth Mover's Distance (EMD) and Gaussian Mixture Models (GMM) [18] is used to detect changes and obtain a segmented point cloud representing those changes. Finally the 3D shape of the object is retrieved using a superquadric approximation to the point cloud.

A. Mapping

Several tests in the mapping field have been performed. In one of them, we have used the Chen-Medioni (point-to-plane) framework for ICP (Iterative Closest Point) [16]. Having a collection of points (p_i, q_i) with normals n_i this algorithm tries to determine the optimal rotation and translation to be applied to the first collection of points, i.e. p_i to bring them into alignment with the second q_i . It obtains a rotation R and translation t trying to minimize the alignment error:

$$\varepsilon = \sum_i [(Rp_i + t - q_i) \cdot n_i]^2 \quad (12)$$

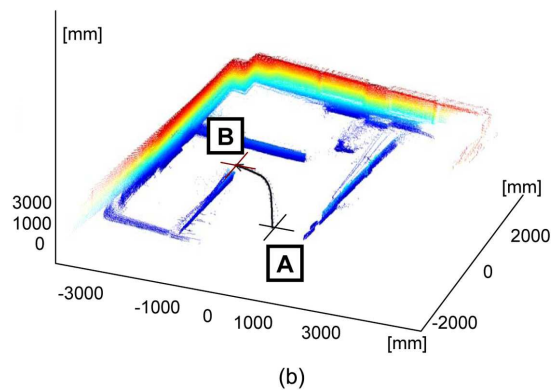
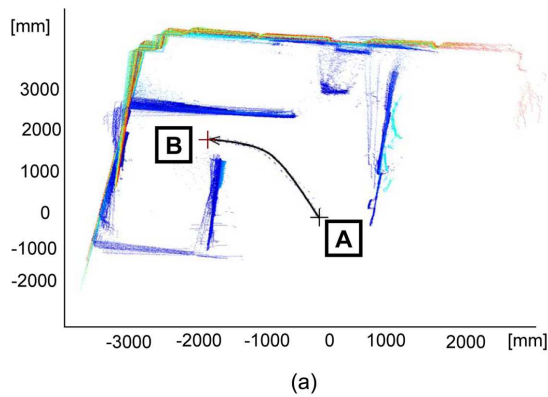


Fig. 7: ICP Scan Matching using 24 different scans and moving from A to B an angle of $\beta \leq 0.2 \text{ rad}$ per scan.

We have obtained 24 different scans moving the robot around the lab and rotating it an angle $\beta \leq 0.2 \text{ rad}$. Data results from the scan matching are shown from two different points of view in Fig. 7a and in Fig 7b. Points A and B from Fig 7 correspond to start and ending points of the scan, respectively. The Scan Matching has been colored on the values of the Y axis (height) in order to make it more visually understandable. Results seem to be quite real and accurate.

After this 24 scans we started increasing the rotation between our scans over 0.2 rad ($\beta \geq 0.2 \text{ rad}$). Then we experienced some error on the applied ICP algorithm, as shown in Fig. 8. It can be seen how walls of our scan start to get deviated as we start increasing the angle of the movements. This could be due to the fact that the quality of alignment obtained by ICP depends heavily on choosing good pairs for corresponding points in the two cloud point [17]. Again, points A and B from Fig 8 correspond, respectively, to start and ending point of the complete scan.

Since ICP does not allow our robot enough freedom in its movements, we have performed a mapping implementation using odometry. To obtain the needed odometry information we have used the *differentialRobot* component that our frame-

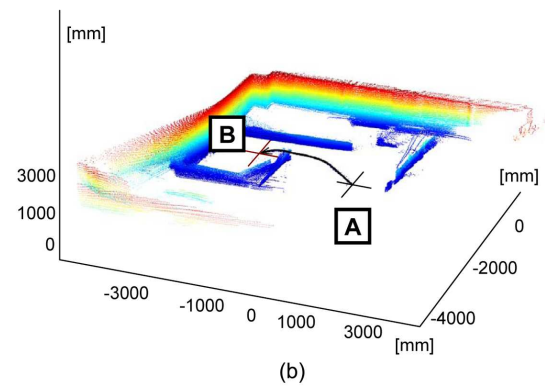
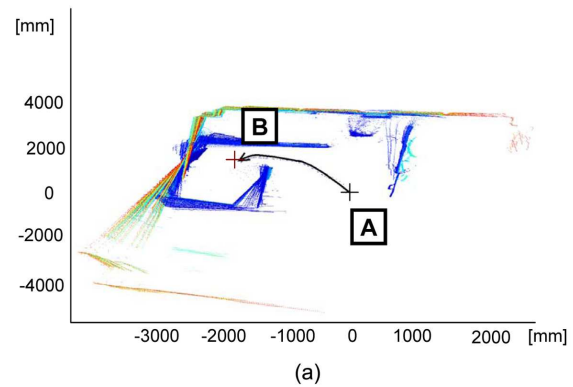


Fig. 8: ICP Scan Matching using 31 different scans and moving from A to B increasing the angle over $\beta \geq 0.2 \text{ rad}$.

work *Robocomp* provides (see (2) in Fig. 4). We have done 31 scans rotating an angle of $\beta = 0.2 \text{ rad}$ one of our Autonomous Mobile Robots (labeled A on Fig. 9). This movement specially confuses the ICP algorithm, and makes it almost impossible to make the match without using odometry. Therefore, making use of the obtained points, combined with the odometry, we have performed the matching showed on Fig. 9. As you can see results are much better than the ones we obtained with ICP. Still an accumulated error on the final scan is showed by the red square, labeled b in Fig. 9b. This, as further explained on Section V, could be solved using for the mapping, along with the 3D data, the texture information our system provides.

B. Novelty Detection and 3D Shape Retrieval based on Gaussian Mixture Models and superquadrics

The second test we have performed on our system is in the field of novelty detection and 3D shape retrieval. We have used the Gaussian Mixture Model for novelty detection and superquadrics for the 3D shape retrieval [18]. There are some main steps in the algorithm: the multi-scale sampling to reduce computation burden; change detection based on Earth's Mover Distance over the point cloud selections of

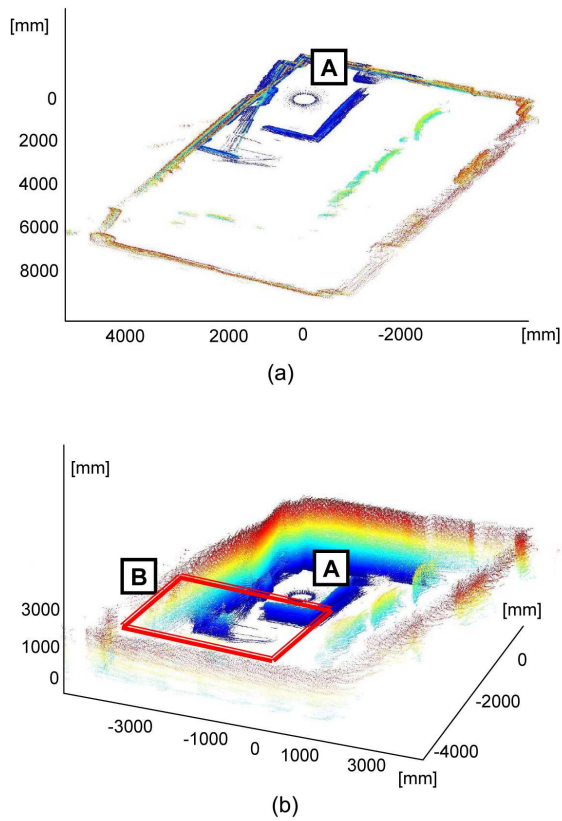


Fig. 9: Data Obtained from a scan in our lab using odometry. Point A shows where the robot has rotated and the red square the accumulated odometry error.

the Gaussian Mixture Model and a 3D shape retrieval of the detected changes using superquadrics.

To perform our experiment we have used our system to scan an empty room in our lab. Then a box have been added to the scene in order to introduce a novelty on the environment. Fig. 10a shows the simplified point cloud (in black) and the correspondent Gaussians associated to the big concentration of points on the empty room, Fig. 10b shows the scene containing the novelty, labeled as (1). Then Gaussians from the first scan are compared to those on the second one and matched using the Earth Mover's Distance EMD. The novelty usually shows up as an unmatched Gaussian as it is the only thing that wasn't there before (labeled (1) in Fig. 10c).

After selecting the correspondent novelty we have retrieved a superquadric and place it on the same place as the box was. The idea of this is to retrieve the 3D shape of the object that was representing the obtained point cloud, in this case a box. Fig. 10d shows a superquadric corresponding to the novelty detected by the previous steps of the algorithm.

This shape retrieval is important in order to provide the mobile robot with a geometric idea of the objects it is facing.

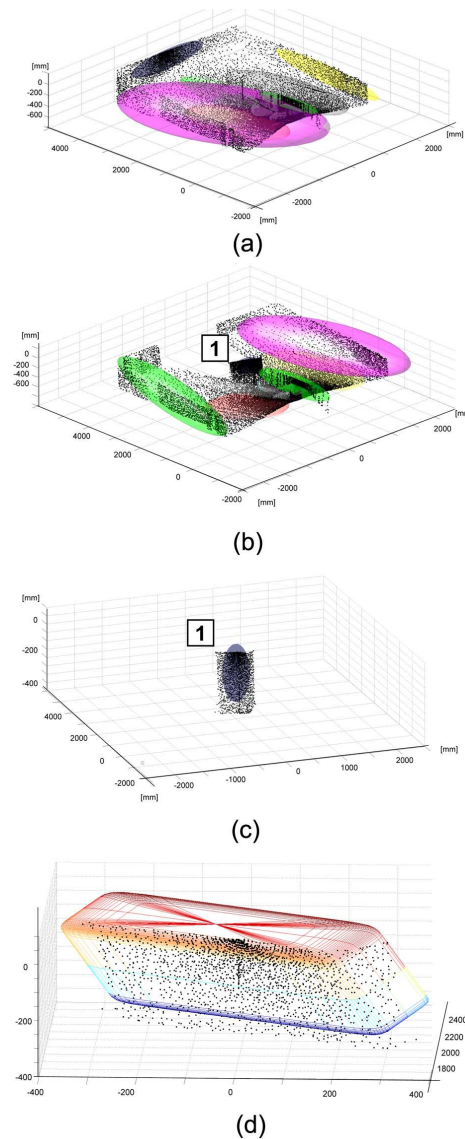


Fig. 10: (a) Simplified point cloud and associated Gaussians of empty room (b) Simplified point cloud and associated Gaussians of room with a novelty (c) Selected novelty on the second scan and associated point cloud (d) Retrieved superquadric from the novelty selected point cloud.

The GMM and superquadrics approach tested here seems quite interesting and challenging. However in our opinion further works are needed since it seems still highly dependable on thresholds and too slow with large datasets, at least if it wants to be used for real-time processing.

V. CONCLUSIONS AND FUTURE WORK

This paper has presented the construction of a 3D sensing system for Autonomous Mobile Robots consisting in a Camera, a 3D LRF and an embedded system. It is intended to be small and power efficient without giving up performance. We have tested our system retrieved data with two different algorithms obtaining promising results. Certainly the step motor combined with the 2D LRF constitutes a solid alternative to those expensive commercial solutions. Data is acquired accurately and fast enough while keeping the low-cost and autonomous requirements. Also, texture information is properly attached to the points coming from the LRF in order to get more information for data processing algorithms. The public interface provided by the communication middleware makes the whole system become a real hardware component, accessible externally through the network. The API offered by the laser component, provides methods to actively scan the world with variable precision.

For the mapping, scan matching using ICP and odometry has been performed. Results are good but still depend on the point cloud provided for ICP, making the algorithm sensible to high changes between scans. Making use of texture information [19] or even making it a real-time mapping with small changes between consecutive scans could constitute a good upgrade for this experiment. In the novelty detection and 3D shape retrieval field the used algorithms (GMM and superquadrics) seemed promising. However, we found some trouble detecting certain point clouds, probably because these algorithms seem highly sensible to thresholds. They, also where very intensive in CPU and memory. Again adding texture information to the algorithm might be a choice, although this could make the application not suitable for real-time use due to a high CPU load. The 2D LRF makes scans at a speed up to 25ms, although our system still retrieves data at a speed of 50ms. This is caused by the embedded processor's CPU whose limited performance still constitutes a bottleneck. New and more powerful ARM processors with multicore architecture have already been announced by Texas Instruments. They are expected for the next few months and will probably solve this problem.

Finally, texture information is attached to the laser through the manual calibration process from section III-B2, another interesting improvement could be to develop an automatic way of finding these camera-laser data correspondences.

ACKNOWLEDGMENTS

This work has partly been supported by grants PRI09A037 and PDT09A044, from the Ministry of Economy, Trade

and Innovation of the Extremaduran Government, by grant TSI-020301-2009-27 from the Spanish Ministry of Industry, Tourism and Commerce, by grant IPT-430000-2010-002 from the Spanish Ministry of Science of Innovation and by FEDER funds from the European Community.

REFERENCES

- [1] N. Ayache and F. Lustman, "Trinocular stereo vision for robotics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 1, pp. 7385, 1991.
- [2] J. Meguro, J. Takiguchi, Y. Amano, and T. Hashizume, "3D reconstruction using multibaseline omnidirectional motion stereo based on gps/dead-reckoning compound navigation system," *Int. Journal of Robotics Research*, vol. 26, pp. 625636, 2007.
- [3] H. P. Moravec, "Robot spatial perception by stereoscopic vision and 3D evidence grids," *CMU Robotics Institute Technical Report CMU-RI-TR-96-34*, 1996.
- [4] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IEEE Workshop on Stereo and Multi-Baseline Vision*, 2001.
- [5] L.J. Manso, P. Bachiller, P. Bustos, P. Nuñez, R. Cintas and L. Calderita. "RoboComp: a Tool-based Robotics Framework". *In Proc. of Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAP)*. Pages 251-262. 2010.
- [6] Trucco, Emanuele and Verri, Alessandro., "Introductory techniques for 3D Computer Vision", Prentice-Hall Inc, 1998.
- [7] Wulf, Oliver and Wagner, Bernardo, "Fast 3D Scanning Methods for Laser Measurement Systems", *Proceedings of the International Conference on Control Systems and Computer Science*, 2003.
- [8] Davis, J.; Chen, X.; , "A laser range scanner designed for minimum calibration complexity," *Proceedings. Third International Conference on 3-D Digital Imaging and Modeling*, 2001, vol., no., pp.91-98, 2001
- [9] Vidal-Calleja, T. and Davison, A.J. and Andrade-Cetto, J. and Murray, D.W., "Active control for single camera SLAM", *Proceedings IEEE International Conference on Robotics and Automation*, 2006.
- [10] Furukawa, Yasutaka; Curless, Brian; Seitz, Steven M.; Szeliski, Richard; , "Reconstructing building interiors from images," *IEEE 12th International Conference on Computer Vision*, 2009, pp.80-87.
- [11] Aliakbarpour, H.; Dias, J.; , "IMU-Aided 3D Reconstruction Based on Multiple Virtual Planes," *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2010, pp.474-479.
- [12] Bosse, Michael; Zlot, Robert; , "Continuous 3D scan-matching with a spinning 2D laser," *International Conference on Robotics and Automation*, 2009. ICRA '09. pp.4312-4319, 12-17.
- [13] Ryde, J.; Huosheng Hu; , "3D Laser range scanner with hemispherical field of view for robot navigation," *International Conference on Advanced Intelligent Mechatronics*, 2008. AIM 2008. IEEE/ASME, vol., no., pp.891-896.
- [14] Hartmut Surmann and Kai Lingemann and Andreas Nchter and Joachim Hertzberg, "A 3D laser range finder for autonomous mobile robots," *Proceedings of the 32nd International Symposium on Robotics*, 2001 pp. 153-158.
- [15] Ryde, J.; , "An inexpensive 3D scanner for indoor mobile robots," *International Conference on Intelligent Robots and Systems, (IROS) 2009*. IEEE/RSJ, pp.5185-5190.
- [16] Chen, Y.; Medioni, G.; , "Object modeling by registration of multiple range images," *Proceedings IEEE International Conference on Robotics and Automation*, 1991. pp.2724-2729 vol.3.
- [17] Gelfand, N.; Ikemoto, L.; Rusinkiewicz, S.; Levoy, M.; , "Geometrically stable sampling for the ICP algorithm," *Proceedings. Fourth International Conference on 3-D Digital Imaging and Modeling*, 2003. 260- 267.
- [18] Drews, P.; Nuñez, P.; Rocha, R.; Campos, M.; Dias, J.; , "Novelty detection and 3D shape retrieval using superquadrics and multi-scale sampling for autonomous mobile robots," *International Conference on Robotics and Automation (ICRA)*, 2010 IEEE, pp.3635-3640.
- [19] Ji Hoon Joung; Kwang Ho An; Jung Won Kang; Myung Jin Chung; Wonpil Yu; "3D environment reconstruction using modified color ICP algorithm by fusion of a camera and a 3D laser range finder," *International Conference on Intelligent Robots and Systems*, 2009. IEEE/RSJ , pp.3082-3088.
- [20] The primesensor Technology, <http://www.primesense.com/>

Chapter 3

An Incremental Hybrid Approach to Indoor Modeling

An incremental hybrid approach to indoor modeling

Marco A. Gutiérrez Pilar Bachiller Luis J. Manso Pablo Bustos Pedro Núñez
Department of Computer and Communication Technology, University of Extremadura, Spain

Abstract—Most of mobile robots basic functions are highly dependent on a model of their environment. Proper modeling is crucial for tasks such as local navigation, localization or route planning. This paper describes a novel solution for building models of indoor environments. We use a 3D Laser on a mobile robot to scan the surroundings and obtain sensing information. While the robot explores the environment, perceived points are clustered forming models of rooms. Room modeling is solved using a new variation of the Hough transform. The result of the modeling process is a topological graph that represents the rooms (nodes) and their connecting doors (edges). Each node in this representation contains the metric parametrization of the room model. Using this basic metric information, robots do not need to maintain in parallel a metric map of the environment. Instead, this metric map can be totally or partially built from the topological representation whenever it is necessary. To test the approach we have carried out a modeling experiment of a real environment, obtaining promising results.

Index Terms—Environment modeling, hybrid representation, active exploration, 3D laser scanning.

I. INTRODUCTION

In recent years, the problem of map building has become an important research topic in the field of robotics. A wide variety of proposals use dense metric maps to represent the robot environment. While a metric representation is necessary for solving certain robot tasks, many others require a more qualitative organization of the environment. Topological maps provide such a qualitative description of the space and constitute a good support to the metric information.

Several approaches on mobile robotics propose the use of topological representation to complement the metric information of the environment. In [16] it is proposed to create off-line topological graphs by partitioning metric maps into regions separated by narrow passages. In [14] the environment is represented by a hybrid topological-metric map composed of a set of local metric maps called *islands of reliability*. [17] describes the environment using a global topological map that associates places which are metrically represented by infinite lines belonging to the same places. [18] constructs a topological representation as a route graph using Voronoi diagrams. In [19] the environment is represented by a graph whose nodes are crossings (corners or intersections). [9] organizes the information of the environment in a graph of planar regions. [3] proposes an off-line method that builds a topological representation, whose nodes correspond to rooms, from a previously obtained metric map.

In this paper, we propose a novel incremental modeling method that provides a hybrid topological/metric representation of indoor environments. Our approach improves the current state of the art in several aspects. Firstly, as in [3], the topological space is represented by a graph whose

nodes encode rooms and whose edges describe connections between rooms. However, in our proposal, the topological representation is incrementally built from the local information provided by the sensor. This means that no global metric map is needed to extract a topological description of the environment. In addition, instead of maintaining in parallel a dense metric map, each topological node contains a minimal set of metric information that allows building a map of a part of the environment when it is needed. This approach reduces drastically the computation the robot must perform to maintain an internal representation of its surroundings. In addition, it can be very helpful for solving certain tasks in an efficient way, such as global navigation or self-localization.

We also present a new variation of the Hough transform used for room modeling. Under the rectangularity assumption, this method provides the geometrical parametrization of a room from a set of points. It is shown how this proposal improves the results compared to other approaches.

The rest of the paper is organized as follows. In section II, an overview of the approach is presented. Section III describes the modeling method that provides a description of the environment in terms of rooms and their connections. Along section IV, the process for creating the proposed hybrid representation is detailed. Experimental results in a real environment are presented in section V. Finally, section VI summarizes the main conclusions of our work.

II. OVERVIEW OF THE APPROACH

As a first approach towards environment modeling, we focus on indoor environments composed by several rooms connected through doors. Rooms are considered approximately rectangular and contain several objects on the floor.

To solve the low-level perceptual process, we have equipped one of our mobile robots with a motorized 2D LRF (Laser Range Finder). Stereo cameras, static 2D LRFs and 3D LRFs constitute an alternative for this purpose. It is possible nowadays to use stereo vision or even the very popular RGB-D primesense sensor[1] to retrieve depth from images and, in consequence, be able to map the robot surroundings. However most of these vision studied techniques are performed under almost ideal circumstances. Uniformly colored surfaces or light variations are some of the problems these solutions might face. In addition, compared to LRF performance, sensors like RGB-D get small fields of view and low depth precision (3cm in 3m scan). LRFs constitute a strong alternative to these sensors, especially when facing not ideal environments. A wide variety of this kind of sensors have become lately available: point range finders, 2D LRFs and 3D LRFs. 3D LRFs seem promising, but their high cost makes them in

practice less usable than other sensors. Motorized 2D LRFs have arisen as a good solution combining important advantages in relation to other sensors for this applications.

To cover the whole 3D spectrum, the 2D LRF is attached to a step motor that makes it scan the whole hemisphere in front of the mobile robot. The resolution is made to be directly dependent on the scanning speed and the robot is able to adjust the related parameters accordingly to its needs. During the exploration, perceived points are stored in a 3D occupancy grid that constitutes a discrete representation of a particular zone of the environment. This occupancy grid is locally used, so, when the robot gets into a new room, the grid is reseted. Each cell of this grid contains the certainty degree about the occupancy of the corresponding volume of the environment. The certainty increases if a point is perceived over time in the same position and decreases when the point vanishes from the space covered by the sensor. Thus, stable points produce higher occupancy values than unstable ones. Cells with a high certainty degree are used for detecting a room model fitting the set of perceived points. Once the model of the current room can be considered stable, it is stored in an internal representation that maintains topological and metric information of the environment.

In this representation, the environment is described as an undirected graph whose vertices represent the different explored rooms (see figure 1). An edge linking two vertices expresses the existence of a door that connects two rooms. This is a very useful representation for the robot to effectively move around man-made environments. For instance, the robot could analyze the graph to obtain the minimum path connecting any two rooms. Moreover, this representation can be extended using recursive descriptions to express more complex world structures like buildings. Thus, a building could be represented by a node containing several interconnected subgraphs. Each subgraph would represent a floor of the building and contain a description of the interconnections between the different rooms and corridors in it.

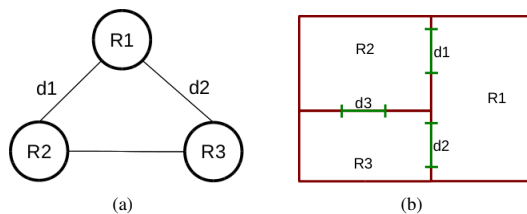


Fig. 1. Topological representation (a) of an environment (b) composed by three intercommunicated rooms.

Using this topological graph, a minimal set of metric information is maintained. Specifically, each vertex of the graph stores the parametric description of the corresponding room and its doors. Using this basic metric information, the robot does not need to maintain in parallel a metric map of the environment. Instead, a total or partial metric representation can be recovered whenever it is necessary from the topological one.

To deal with the uncertainty derived from odometric and sensor errors, we follow a non simultaneous approach to localization and modeling. Modeling errors are minimized

by performing specific actions directed to quickly obtain measures around the robot. Once a model is fixed, errors in the following models, i.e. adjacent rooms, are canceled by applying geometric restrictions. Finally, detected loop closings are used through a global minimization procedure to further reduce modeling errors. Localization errors are canceled by continuously rebuilding a new model that is compared to the current one. Using an estimated rectangular model instead of the raw measures is an effective procedure to update the position of the robot relative to the current model, since it is less sensitive to wrong measurements. As long as the rectangular hypothesis is coherent with the real world around the robot, these methods work robustly in real situations. More details on this are given on section IV.

III. ROOMS AND DOORS MODELING

Since rooms are assumed to be rectangular and its walls perpendicular to the floor, the problem of modeling a room from a set of points can be treated as a rectangle detection problem using the projections of those points onto the floor plane. Several rectangle detection techniques can be found in the literature [5, 6, 15]. Most of them are based on a search in the 2D point space (for instance, a search in the edge representation of an image) using line primitives. These methods are computationally expensive and can be very sensitive to noisy data. In order to solve the modeling problem in an efficient way, we propose a new rectangle detection technique based on a search in the parameter space using a variation of the Hough Transform [13, 2].

For line detection, several variations of the Hough Transform have been proposed [8, 11]. The extension of the Hough Transform for rectangle detection is not new. [20] proposes a *Rectangular Hough Transform* used to detect the center and orientation of a rectangle with known dimensions. [4] proposes a *Windowed Hough Transform* that consists of searching rectangle patterns in the Hough space of every window of suitable dimensions of an image.

Our approach for rectangle detection uses a 3D version of the Hough Transform that facilitates the detection of segments instead of lines. This allows considering only those points that belong to the contour of a rectangle in the detection process, which is very important to obtain reliable results. For instance, consider the 2D view of the occupancy grid that is shown in figure 2(a). In this situation, the robot has perceived all the walls and objects of the room it is located and, partially, two walls of the adjoining room. Figures 2(b) and 2(c) show the results of the rectangle detection process using variations of the Hough transform based on lines and segments, respectively. The lined region of both figures corresponds to the detected rectangle. As it can be observed, a method based on lines, as the one proposed in [4], considers points that can be situated outside the rectangle, leading sometimes to wrong results. Nevertheless, the proposed variation of the Hough transform takes into account only those points belonging to the four segments of the rectangle providing always the best rectangle pattern.

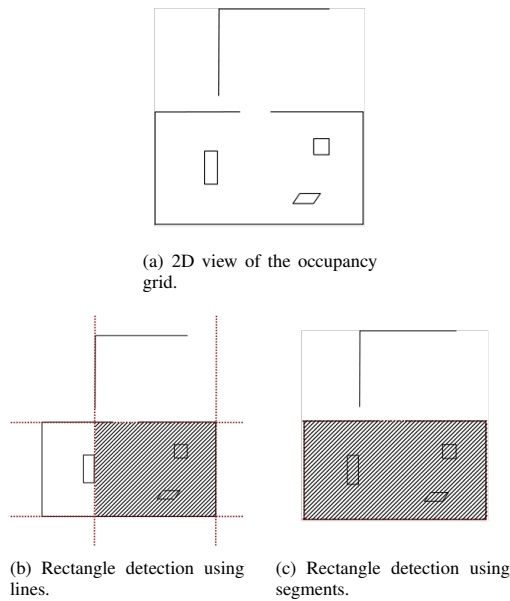


Fig. 2. Rectangle detection using the variation of the Hough transform based on lines (b) and the proposed one based on segments (c).

A. Room detection

In our room detection method, the Hough space is parameterized by (θ, d, p) , being θ and d the parameters of the line representation ($d = x \cos(\theta) + y \sin(\theta)$) and $|p|$ the length of a segment in the line. For computing p it is assumed that one of the extreme points of its associated segment is initially fixed and situated at a distance of 0 to the perpendicular line passing through the origin. Under this assumption, being (x, y) the other extreme point of the segment, its *signed* length p can be computed as:

$$p = x \cos(\theta + \pi/2) + y \sin(\theta + \pi/2) \quad (1)$$

Using this representation, any point (x, y) contributes to those points (θ, d, p) in the Hough space that verifies:

$$d = x \cos(\theta) + y \sin(\theta) \quad (2)$$

and

$$p \geq x \cos(\theta + \pi/2) + y \sin(\theta + \pi/2) \quad (3)$$

Equation 2 represents every line intersecting the point as in the original Hough Transform. The additional condition expressed by equation 3 limits the point contribution to those line segments containing the point. This allows computing the total number of points included in a given segment. For instance, given a segment with extreme points $V_i = (x_i, y_i)$ and $V_j = (x_j, y_j)$ and being H the 3D Hough space, the number of points that belongs to the segment, which is denoted as $H_{i \leftrightarrow j}$, can be computed as:

$$H_{i \leftrightarrow j} = |H(\theta_{i \leftrightarrow j}, d_{i \leftrightarrow j}, p_i) - H(\theta_{i \leftrightarrow j}, d_{i \leftrightarrow j}, p_j)| \quad (4)$$

where $\theta_{i \leftrightarrow j}$ and $d_{i \leftrightarrow j}$ are the parameters of the common line to both points and p_i and p_j are the *signed* lengths of

the two segments with non-fixed extreme points V_i and V_j , respectively, according to equation 1.

Since a rectangle is composed of four segments, the 3D Hough space parameterized by (θ, d, p) allows computing the total number of points included in the contour of the rectangle. Thus, considering a rectangle expressed by its four vertices $V_1 = (x_1, y_1)$, $V_2 = (x_2, y_2)$, $V_3 = (x_3, y_3)$ and $V_4 = (x_4, y_4)$ (in clockwise order), the number of points of its contour, denoted as H_r , can be computed as:

$$H_r = H_{1 \leftrightarrow 2} + H_{2 \leftrightarrow 3} + H_{3 \leftrightarrow 4} + H_{4 \leftrightarrow 1} \quad (5)$$

Considering the restrictions about the segments of the rectangle and using the equation 4, each $H_{i \leftrightarrow j}$ of the expression 5 can be rewritten as follows:

$$H_{1 \leftrightarrow 2} = |H(\alpha, d_{1 \leftrightarrow 2}, d_{4 \leftrightarrow 1}) - H(\alpha, d_{1 \leftrightarrow 2}, d_{2 \leftrightarrow 3})| \quad (6)$$

$$H_{2 \leftrightarrow 3} = |H(\alpha + \pi/2, d_{2 \leftrightarrow 3}, d_{1 \leftrightarrow 2}) - H(\alpha + \pi/2, d_{2 \leftrightarrow 3}, d_{3 \leftrightarrow 4})| \quad (7)$$

$$H_{3 \leftrightarrow 4} = |H(\alpha, d_{3 \leftrightarrow 4}, d_{2 \leftrightarrow 3}) - H(\alpha, d_{3 \leftrightarrow 4}, d_{4 \leftrightarrow 1})| \quad (8)$$

$$H_{4 \leftrightarrow 1} = |H(\alpha + \pi/2, d_{4 \leftrightarrow 1}, d_{3 \leftrightarrow 4}) - H(\alpha + \pi/2, d_{4 \leftrightarrow 1}, d_{1 \leftrightarrow 2})| \quad (9)$$

being α the orientation of the rectangle and $d_{i \leftrightarrow j}$ the normal distance from the origin to the straight line defined by the points V_i and V_j .

Since H_r expresses the number of points in a rectangle r defined by $(\alpha, d_{1 \leftrightarrow 2}, d_{2 \leftrightarrow 3}, d_{3 \leftrightarrow 4}, d_{4 \leftrightarrow 1})$, the problem of obtaining the best rectangle given a set of points can be solved by finding the combination of $(\alpha, d_{1 \leftrightarrow 2}, d_{2 \leftrightarrow 3}, d_{3 \leftrightarrow 4}, d_{4 \leftrightarrow 1})$ that maximizes H_r . This parametrization of the rectangle can be transformed into a more practical representation defined by the five-tuple (α, x_c, y_c, w, h) , being (x_c, y_c) the central point of the rectangle and w and h its dimensions. This transformation can be achieved using the following expressions:

$$x_c = \frac{d_{1 \leftrightarrow 2} + d_{3 \leftrightarrow 4}}{2} \cos(\alpha) - \frac{d_{2 \leftrightarrow 3} + d_{4 \leftrightarrow 1}}{2} \sin(\alpha) \quad (10)$$

$$y_c = \frac{d_{1 \leftrightarrow 2} + d_{3 \leftrightarrow 4}}{2} \sin(\alpha) + \frac{d_{2 \leftrightarrow 3} + d_{4 \leftrightarrow 1}}{2} \cos(\alpha) \quad (11)$$

$$w = d_{2 \leftrightarrow 3} - d_{4 \leftrightarrow 1} \quad (12)$$

$$h = d_{3 \leftrightarrow 4} - d_{1 \leftrightarrow 2} \quad (13)$$

In order to compute H_r , the parameter space H is discretized assuming the rank $[-\pi/2, \pi/2]$ for θ and $[d_{min}, d_{max}]$ for d and p , being d_{min} and d_{max} the minimum and maximum distance, respectively, between a line and the origin. Taking some sample step for each parameter and being G the 3D occupancy grid and τ the minimum occupancy value to consider a non-empty region of the environment, the proposed method for room modeling can be summarized in the following steps:

- 1) Initialize all the cells of the discrete Hough space H to 0.
- 2) For each cell, $G(x_d, y_d, z_d)$, such that $G(x_d, y_d, z_d).occupancy > \tau$:
 Compute the real coordinates (x, y) associated to the cell indexes (x_d, y_d) .
 For $\theta_d = \theta_{dMin} \dots \theta_{dMax}$:
 a) Compute the real value θ associated to θ_d .

- b) Compute $d = x \cos(\theta) + y \sin(\theta)$.
 - c) Compute the discrete value d_d associated to d .
 - d) Compute $p = x \cos(\theta + \pi/2) + y \sin(\theta + \pi/2)$.
 - e) Compute the discrete value p_d associated to p .
 - f) For $p'_d = p_d \dots d_{dMax}$: increment $H(\theta_d, d_d, p'_d)$ by 1.
- 3) Compute $\text{argmax}_r H_r(\alpha, d_{1 \leftrightarrow 2}, d_{2 \leftrightarrow 3}, d_{3 \leftrightarrow 4}, d_{4 \leftrightarrow 1})$.
 - 4) Obtain the rectangle $r = (\alpha, x_c, y_c, w, h)$ using equations 10, 11, 12 and 13.

As it can be observed, the height of walls is only taken into account through histogram contributions. This is because walls correspond to 2D segment with higher histogram values than any other plane perpendicular to the floor. Thus, it is not necessary to explicitly consider the height of points in the room detection method.

Even though this method is computationally expensive, in practice, its complexity can be significantly reduced in two ways. Firstly, instead of computing H from the whole occupancy grid, it can be updated using only those cells whose occupancy state has changed. In addition, it is not necessary to apply step 3 over the entire parameter space, since only rectangles of certain dimensions are considered rooms. Thus, it is assumed a specific rank of w and h that limits the search to those values of $d_{1 \leftrightarrow 2}$, $d_{2 \leftrightarrow 3}$, $d_{3 \leftrightarrow 4}$ and $d_{4 \leftrightarrow 1}$ fulfilling that rank.

B. Door detection

The proposed 3D Hough space is also used for door detection. Doors are free passage zones that connect two different rooms, so they can be considered as empty segments of the corresponding room rectangle (i.e. segments without points). Taking this into account, once the room model is obtained, doors can be detected by analyzing each wall segment in the 3D Hough space. Therefore, for each segment of the rectangle, defined by V_i and V_j , two points $D_k = (x_k, y_k)$ and $D_l = (x_l, y_l)$ situated on the inside of that segment constitute a door segment if they verify:

$$H_{k \leftrightarrow l} = |H(\theta_{i \leftrightarrow j}, d_{i \leftrightarrow j}, p_k) - H(\theta_{i \leftrightarrow j}, d_{i \leftrightarrow j}, p_l)| = 0 \quad (14)$$

being $\theta_{i \leftrightarrow j}$ and $d_{i \leftrightarrow j}$ the parameters of the straight line defined by V_i and V_j and p_k and p_l the *signed* lengths of the segments for D_k and D_l :

$$p_k = x_k \cos(\theta_{i \leftrightarrow j} + \pi/2) + y_k \sin(\theta_{i \leftrightarrow j} + \pi/2) \quad (15)$$

$$p_l = x_l \cos(\theta_{i \leftrightarrow j} + \pi/2) + y_l \sin(\theta_{i \leftrightarrow j} + \pi/2) \quad (16)$$

Assuming $p_i \leq p_k < p_l \leq p_j$ and a minimum length l for each door segment, the door detection process can be carried out by verifying equation 14 for every pair of points between V_i and V_j , such that $p_l - p_k \geq l$. Starting from the discrete representation of the Hough space, this process can be summarized in the following steps:

- 1) Compute the discrete value θ_d associated to θ_{i-j} .
- 2) Compute the discrete value d_d associated to d_{i-j} .
- 3) Compute the discrete value p_{di} associated to p_i .
- 4) Compute the discrete value p_{dj} associated to p_j .

- 5) Compute the discrete value l_d associated to l (minimum length of doors).
- 6) $p_{dk} \leftarrow p_{di}$
- 7) While $p_{dk} \leq p_{dj} - l_d$:
 - a) $p_{dl} \leftarrow p_{dk} + 1$
 - b) While $p_{dl} < p_{dj}$ and $|H(\theta_d, d_d, p_{dk}) - H(\theta_d, d_d, p_{dl})| = 0$: $p_{dl} \leftarrow p_{dl} + 1$
 - c) If $p_{dl} - p_{dk} > l_d$:
 - i) Compute the real value p_k associated to p_{dk} .
 - ii) Compute the real value p_l associated to $(p_{dl} - 1)$.
 - iii) Compute the door limits D_k and D_l from p_k and p_l .
 - iv) Insert the new door segment with extreme points D_k and D_l to the list of doors.
 - d) $p_{dk} \leftarrow p_{dl}$

IV. INCREMENTAL MODELING OF THE ENVIRONMENT

Building topological maps requires to endow the robot not only with modeling skills, but also with the ability to actively explore the environment. Exploration plays an important role in our proposal because the robot must make sure that each room model corresponds to a real room before leaving it behind. For this reason the robot must scan the whole space surrounding it to take correct decisions about the current model.

In our system, the exploration task is driven by the 3D local grid. When a room model is detected from the set of points stored in the grid, the robot must verify it by scanning the unexplored zones inside the estimated room model. Depending on its occupancy value, the cells of the grid are labeled as *occupied*, *empty* and *unexplored*. Thus, by analyzing the grid, the robot can direct its sensor towards new places and retrieve the necessary information to get a reliable model. At this point, the benefits of using a long range sensor become clear. Few movements of the robot are needed to cover the whole space around it and, in consequence, the modeling process is less sensitive to odometric errors.

Once the local space around the robot has been completely explored, the current room model is inserted as a node in the graph representing the topological space and the robot gets out of the room to model new places. Each node in the graph stores the geometric parametrization of the room and its doors. The center and orientation of the room are used to form a reference frame (F_r) that expresses the location of the room in relation to a global reference frame (F_w). Thus, being $r = (\alpha, x_c, y_c, w, h)$ the rectangle that models a given room, the transformation matrix (T_r) that relates F_r with F_w is defined as:

$$T_r = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & x_c \\ \sin(\alpha) & \cos(\alpha) & 0 & y_c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (17)$$

Using this transformation matrix, any point of the model is expressed in coordinates relative to the room. This way, if the local or global reference frames are modified, points of room models remain unaffected. In addition, the robot can be aware

of errors in its odometric estimation by detecting changes in the room reference frame. These changes come from the estimation of new room models, with the same dimensions than the current one, using the set of recently perceived points. Thus, being $r(i) = (\alpha(i), x_c(i), y_c(i), w, h)$ the room model at instant i and $r(i+1) = (\alpha(i+1), x_c(i+1), y_c(i+1), w, h)$ a new estimation of the room model at $i+1$, changes in the robot pose can be computed using the rotational and translational model deviations of equations 18 and 19.

$$\Delta\alpha = \alpha(i+1) - \alpha(i) \quad (18)$$

$$\Delta t = \begin{pmatrix} x_c(i+1) \\ y_c(i+1) \\ 0 \end{pmatrix} - \begin{pmatrix} \cos(\Delta\alpha) & -\sin(\Delta\alpha) & 0 \\ \sin(\Delta\alpha) & \cos(\Delta\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c(i) \\ y_c(i) \\ 0 \end{pmatrix} \quad (19)$$

This is useful once the model is created for dealing with the problem of robot pose estimation. However, odometric errors are present during the whole modeling process, affecting the resulting representation in two ways. Firstly, when the robot models a new room, the location of the new room could not match its real location. Secondly, odometric errors lead to wrong measurements that cause imperfect estimations of the parameters of rooms and doors.

Though the essence of these two problems is slightly different, both can be detected and corrected using the notion of adjacent rooms. Therefore, if two adjacent rooms, r_1 and r_2 , are communicated by a door, any point of the door is common to both rooms. Assume a door point d_{r_1} viewed from the room r_1 and the corresponding point d_{r_2} of the room r_2 . The metric representation of both rooms would ideally be subject to the following restriction:

$$\|T_{r_2}d_{r_2} - T_{r_1}d_{r_1}\|^2 = 0 \quad (20)$$

When the robot creates a new room model that is adjacent to a previous one, expression 20 determines the need for correcting the new model reference frame. In addition, the deviation between the positions of the common door allows computing how this correction should be applied in order to fulfill the restriction imposed by the expression above.

After an exploration of arbitrary length, if the robot returns to a previously visited room (i.e. a *loop closing* is detected), the non-correspondence between the input and output doors can also be determined using expression 20. In such cases, new corrections must be done in order to cancel the error. However, this error is caused by wrong estimations of room and door models and, in consequence, a reference frame correction will surely not solve the problem. Our solution to these situations is to minimize a global error function defined over the whole metric representation. In our representation of the environment, the global error is defined in terms of deviations between the positions of the doors connecting adjacent rooms. Thus, the error function to minimize can be expressed as:

$$\xi = \sum_{\forall \text{connected}(d_{r_i}^{(n)}, d_{r_j}^{(m)})} \|T_{r_i}d_{r_i}^{(n)} - T_{r_j}d_{r_j}^{(m)}\|^2 \quad (21)$$

being $d_{r_i}^{(n)}$ and $d_{r_j}^{(m)}$ the middle points of a common door expressed in the reference frames of rooms r_i and r_j , respec-

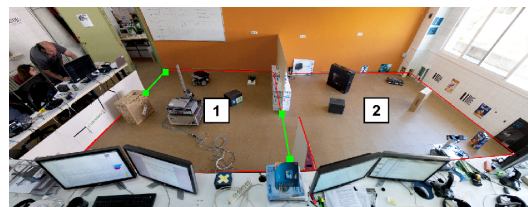
tively, and T_{r_i} and T_{r_j} the transformation matrices of such rooms.

The employed minimization method is based in the Stochastic Gradient Descent [12]. The basic idea [10] is to minimize the global error function by introducing small variations in the parameters of room and door models. These variations are constrained by the uncertainty of the measurement, so *high-confident* parameters remain almost unchanged during the error minimization process. As result, a reliable representation of the environment that maintains the restrictions imposed by the real world is obtained.

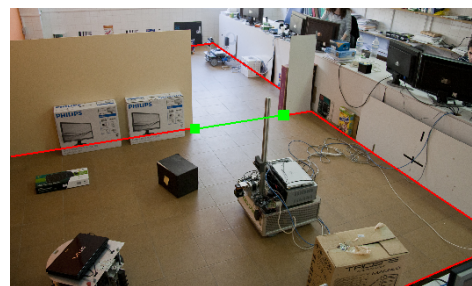
V. EXPERIMENTAL RESULTS

We used one of our custom differential robots to build a model of the environment. It has been equipped with a Hokuyo UTM-30LX 2D LR moved by a step motor. It scans up to 30 meters and a range of 270° at a rate of 25ms per line of scan (700 points). The LRF is positioned in the front of the robot and it performs a roll movement to cover an amplitude of 360° , (see figure 3(b)). The software has been developed using the component oriented robotics framework RoboComp [7].

The experimental results presented in this paper correspond to the modeling of an environment formed by two contiguous rooms with its corresponding doors. Rooms have been made to be similar to any regular room and have been filled with random objects to simulate a real human environment. Figure 3 shows two views of the environment used in this experiment. In these views, the two rooms, labeled as 1 and 2, and the doors have been marked in red and green, respectively.



(a) Frontal view of the scene.



(b) Side view of the scene taken from the left up corner of 3(a).

Fig. 3. Two views of the real scene of the experiment.

We placed our mobile robot in the center of room 1 in order to start our testing. In the first part of the experiment, the robot performs a full 3D scan of the first room. Results

can be seen in figure 4. From its initial location, the robot makes a first scan and obtains an initial model of the room (figures 4(a) and 4(b)). Figure 4(a) shows the regions and the model while figure 4(b) shows only the model. Notice that regions containing points are considered walls (in red) and those without points are considered doors (in green). Using this initial model, the robot tries now to scan the unknown areas. After moving 90° counter clockwise, new points are obtained from a second scan. Figures 4(c) and 4(d) show on regions and model results, respectively. As it can be observed, the accuracy of the model increases with this second scan. The detected rectangle fits the room size but there are still some unknown parts considered as doors. The robot tries to verify whether those regions are real doors or not by moving 180° and performing another scan. After this new scan, the room is perfectly matched by the model, as figures 4(e) and 4(f) show. The room scanning is finished when all regions in the model have been scanned, and further scanning (even using higher resolutions) results in no modifications. Then, the obtained room model is fixed and stored in the topological graph. Notice that, although the model size and doors are properly obtained, the room has been a little mispositioned when compared to reality. This is due to the accumulation of odometric errors during the robot movements. Nevertheless, the relative position of the robot inside the room remains correct and therefore the detection of a new room is not affected by this misplacement.

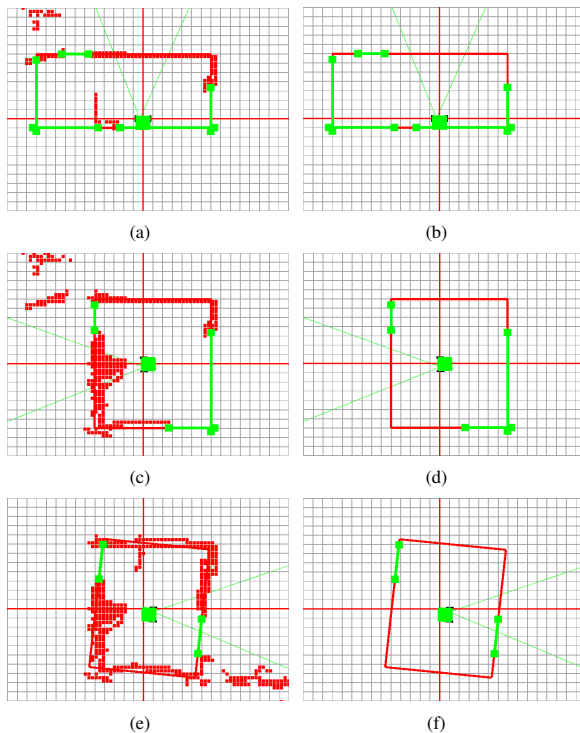


Fig. 4. Modeling process of room 1.

After the model of the first room has been fixed, the robot

uses one of the obtained doors to get to the next room. In the experiment, the robot goes through the door on the right to room 2 and performs a full 3D scan. Figure 5 shows results of the second room scans. Specifically figures 5(a) and 5(b) correspond to the first one. As it can be observed, the model is still not matching the real room because of the existence of big spaces with missing information. Therefore the robot turns and get another scan of one of these places. Figures 5(c) and 5(d) show the model and regions after the second scan in this room. Now the model fits the room but, again there are still unscanned parts. The robot turns 90° again and performs the scan whose regions and resulting model are shown in figures 5(e) and 5(f). After doing all these scans in the second room the final model is obtained. Further scans do not change the model, so it is fixed.

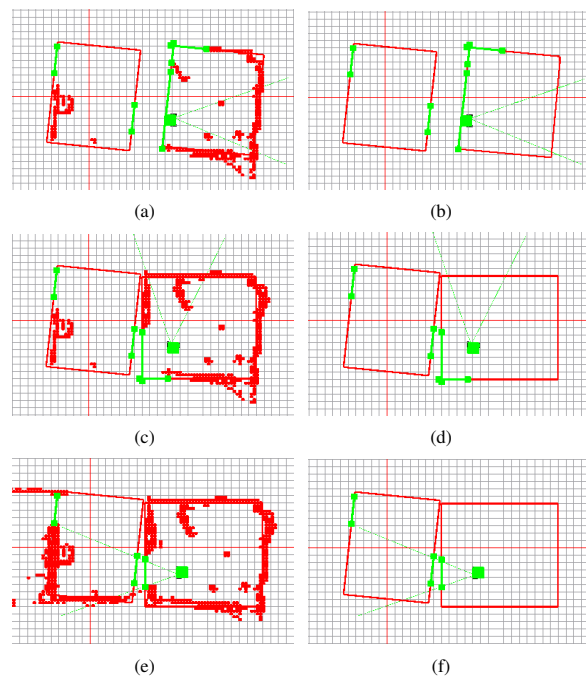


Fig. 5. Modeling process of room 2.

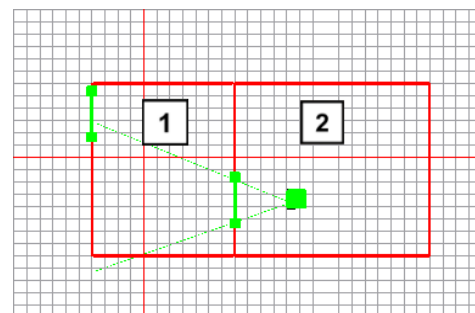


Fig. 6. Final representation obtained after the modeling of the two rooms.

Once the second model has been fixed, the deviation between the two room models (figure 5(f)), caused by odometric errors, is corrected according to the common door restriction (equation 20). Figure 6 shows the result of this correction. Each square of the figure represents an area of $0,27 \times 0,27m^2$. This size corresponds to the sampling step of the Hough space for the parameters d and p . This means that the accuracy of each room model is limited by this sampling step. The real sizes of the two rooms are $3,19 \times 3,78m^2$ (room 1) and $4,20 \times 3,78m^2$ (room 2). The sizes obtained by the modeling process are $2,97 \times 3,78m^2$ (room 1) and $4,05 \times 3,78m^2$ (room 2). As it can be observed, the difference between the representation and the real world is in the range of the permissible error. More accurate models can be obtained by reducing the sampling step of the discrete Hough space.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented an incremental modeling method for building hybrid representations of indoor environments. The proposed representation consists in a topological graph that describes the rooms of the environment and their connections. Each node of the graph represents a room and contains the geometric parametrization of the corresponding room and its doors. This geometric parametrization is the only metric information included in the representation. Using this information, the robot can recover a metric map of a particular zone of the environment when it is needed. Thus, no dense metric map is maintained in parallel to the topological graph. Rooms and doors are modeled using a variation of the Hough Transform that detects segments and rectangle patterns. We have proposed methods for dealing with odometric errors in the creation of new models as well as in loop closings. In addition, a method for robot pose estimation using room models has been presented. Real experiments have been carried out using a mobile robot equipped with a motorized 2D Laser. Results show the accuracy of the modeling process in real environments.

We are working on several improvements of our proposal. In particular, work in order to relax the rectangle assumption is currently in progress. We are also extending the modeling ability of our robots for representing other structures of man-made environments like corridors. Bigger and more complex possible surroundings are also being taken into account.

ACKNOWLEDGMENT

This work has partly been supported by grants PRI09A037 and PDT09A044, from the Ministry of Economy, Trade and Innovation of the Extremaduran Government, and by grants TSI-020301-2009-27 and IPT-430000-2010-2, from the Spanish Government and the FEDER funds.

REFERENCES

- [1] The primesensor technology. <http://www.primesense.com/>.
- [2] R.O. Duda and P.E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15:11–15, January 1972.
- [3] Kwangro Joo, Tae-Kyeong Lee, Sanghoon Baek, and Se-Young Oh. Generating topological map from occupancy grid-map using virtual door detection. In *IEEE Congress on Evolutionary Computation*, pages 1–6, 2010.

- [4] C.R. Jung and R. Schramm. Rectangle detection based on a windowed hough transform. In *Proceedins of the XVII Brazilian Symposium on Computer Graphics and Image Processing*, pages 113–120, 2004.
- [5] D. Lagunovsky and S. Ablameyko. Straight-line-based primitive extraction in grey-scale object recognition. *Pattern Recognition Letters*, 20(10):1005–1014, 1999.
- [6] C. Lin and R. Nevatia. Building detection and description from a single intensity image. *Computer Vision and Image Understanding*, 72(2):101–121, 1998.
- [7] L. Manso, P. Bachiller, P. Bustos, P. Núñez, R. Cintas, and L. Calderita. Robocomp: a tool-based robotics framework. In *Proceedings of the Second international conference on Simulation, modeling, and programming for autonomous robots*, SIMPAR'10, pages 251–262, 2010.
- [8] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.
- [9] E. Montijano and C. Sagues. Topological maps based on graphs of planar regions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1661–1666, October 2009.
- [10] E. Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.
- [11] P.L. Palmer, J. Kittler, and M. Petrou. Using focus of attention with the hough transform for accurate line parameter estimation. *Pattern Recognition*, 27(9):1127–1134, 1994.
- [12] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.
- [13] A. Rosenfeld. Picture processing by computer. *ACM Comput. Surv.*, 1:147–176, September 1969.
- [14] S. Simhon and G. Dudek. A global topological map formed by local metric maps. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1708–1714, 1998.
- [15] W.-B. Tao, J.-W. Tian, and J. Liu. A new approach to extract rectangular building from aerial urban images. In *Signal Processing, 2002 6th International Conference on*, volume 1, pages 143 – 146, aug. 2002.
- [16] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.
- [17] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 44(1):3–14, 2003.
- [18] D. Van Zwynsvoorde, T. Simeon, and R. Alami. Incremental topological modeling using local vorono-like graphs. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and System (IROS 2000)*, volume 2, pages 897–902, 2000.
- [19] F. Yan, Y. Zhuang, and W. Wang. Large-scale topological environmental model based particle filters for mobile robot indoor localization. *Robotics and Biomimetics, IEEE International Conference on*, 0:858–863, 2006.
- [20] Y. Zhu, B. Carragher, F. Mouche, and C.S. Potter. Automatic particle detection through efficient hough transforms. *IEEE Trans. Med. Imaging*, 22(9), 2003.

Chapter 4

A Passive Learning Sensor Architecture for Multimodal Image Labeling: An Application for Social Robots



Article

A Passive Learning Sensor Architecture for Multimodal Image Labeling: An Application for Social Robots

Marco A. Gutiérrez ^{1,*}, Luis J. Manso ¹, Harit Pandya ² and Pedro Núñez ¹

¹ Robotics and Artificial Vision Laboratory, University of Extremadura, 10003 Cáceres, Spain; lmanso@unex.es (L.J.M.); pnuntru@unex.es (P.N.)

² Robotics Research Center, IIIT Hyderabad, 500032 Hyderabad, India; harit.pandya@research.iiit.ac.in

* Correspondence: marcog@unex.es; Tel.: +34-927-257-259

Academic Editor: Vittorio M. N. Passaro

Received: 21 December 2016; Accepted: 8 February 2017; Published: 11 February 2017

Abstract: Object detection and classification have countless applications in human–robot interacting systems. It is a necessary skill for autonomous robots that perform tasks in household scenarios. Despite the great advances in deep learning and computer vision, social robots performing non-trivial tasks usually spend most of their time finding and modeling objects. Working in real scenarios means dealing with constant environment changes and relatively low-quality sensor data due to the distance at which objects are often found. Ambient intelligence systems equipped with different sensors can also benefit from the ability to find objects, enabling them to inform humans about their location. For these applications to succeed, systems need to detect the objects that may potentially contain other objects, working with relatively low-resolution sensor data. A passive learning architecture for sensors has been designed in order to take advantage of multimodal information, obtained using an RGB-D camera and trained semantic language models. The main contribution of the architecture lies in the improvement of the performance of the sensor under conditions of low resolution and high light variations using a combination of image labeling and word semantics. The tests performed on each of the stages of the architecture compare this solution with current research labeling techniques for the application of an autonomous social robot working in an apartment. The results obtained demonstrate that the proposed sensor architecture outperforms state-of-the-art approaches.

Keywords: robot sensors; ambient intelligence sensors; deep learning; object detection; object recognition; word semantics

1. Introduction

The possibilities and applications of autonomous social robots and ambient intelligence systems in our daily activities can be of utmost help. For the case of social robots, the research community has already studied how to perform several kinds of these tasks, such as cloth folding [1], floor vacuuming [2], cooking [3] or even more complex home assistant applications [4]. A basic skill on which these applications heavily rely is the need to identify and locate common household objects. Additionally, the problem of searching and finding objects extends to other automation applications requiring context awareness, e.g., Kidd et al. [5]. The complexity and variety in shape and location of these objects makes their detection one of the most complex skills to develop. Due to this complexity and despite the important advances in computer vision [6], most of the current solutions are still not robust enough to be applied in real household scenarios where clutter and changing environment conditions are common. Therefore, due to the importance of object detection for real household tasks, the development of methods to improve it has become a challenging and interesting research topic.

A property of indoor environments is that their items are usually structured hierarchically (e.g., an object lies on a table which is located in a certain room). Maintaining a representation of the environment and these hierarchical relationships can become extremely helpful when segmenting the environment. Additionally, objects in household environments are often found grouped by categories and placed relatively close according to their use, e.g., kitchen utensils are arranged together in the kitchen and toys are usually stored in a specific location in a certain room. These two properties can be exploited to successfully find and classify objects in indoor environments. Techniques such as cognitive subtraction [7] allow robots to use their world model in order to detect potentially unknown objects. The work presented here takes advantage of these properties and uses them to improve the detection of objects in household environments under real-world conditions.

It has been demonstrated that long-term memory representations have a strong influence on human visual performance [8]. Therefore, taking advantage of previous experience through a long-term memory of a system is another key value that can be added to autonomous systems in order to improve the search of objects. The Passive Learning Sensor Architecture (PLSA) has been developed in order to exploit these advantages and to improve the performance when trying to find objects in large environments. It uses a world representation to filter the input data and focus the attention to those places where the objects of interest could be found. This information is obtained while the system is performing any task, even when it is apparently stopped (see the social robot use case in Figure 1). It is designed to be able to deal with low resolution images or light changes. Processing these images, the system learns the most probable object locations in order to help improve the time taken to find objects in later situations. The main contribution of the PLSA is that it improves the performance of social robots and autonomous systems in the task of finding objects in large environments through the combination of image labeling with word semantics. It uses multimodal information—language semantic information from trained models combined with visual input data—in order to estimate the most likely location for any given object. The system also exploits the existent semantic relationships among objects in the same object container to improve the final results. In the context of this paper, any object which can have objects inside or over it is considered an object container, i.e., a table, a shelf or a cupboard. However, in the use case presented here, a social robot searching for objects in an apartment, all the containers used are tables.

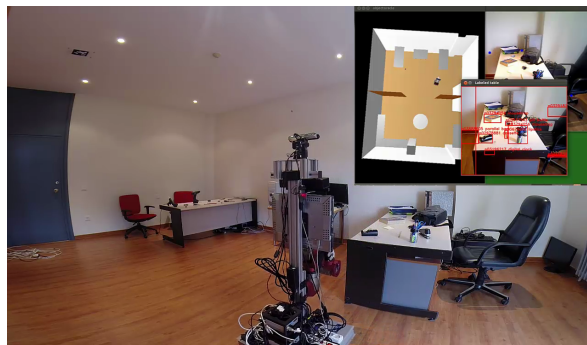


Figure 1. Shelly, a humanoid social robot, exploring an apartment and labeling objects on tables. The visualization of the labeling process is shown in the top right corner.

The pipeline of the architecture is composed of four main steps: a first cognitive attention stage to locate and segment object containers; a segmentation step to select smaller windows to label as objects; a Deep Neural Network (DNN) that is able to label the selected windows; and a final semantic matching step to improve and fine tune labeling results.

The use case of an implementation of PLSA in a social robot searching for objects in a 65 m² apartment is presented for a better understanding. The robot learns from the images acquired while

moving through the apartment during the execution of previous tasks. The PLSA takes advantage of its multimodality and combines these images with language semantic information to make early predictions on possible object locations. This way, the social robot is able to optimize the path to a successful search when searching for objects.

The remainder of this paper is organized as follows. After a brief summary about similar works in the literature in Section 2, Section 3 deeply explains the design of the PLSA. Section 4 describes the experiments that have been run to test the proposed sensor architecture. The conclusions and future lines of work are outlined in Section 5. For an easier understanding of the abbreviations used along the explanation of this work, a reference list is included at the end of the manuscript.

2. State-of-the-Art

The research community has demonstrated a high interest in the problem of active visual object search, however only a few works using semantic information have been published. Rangel et al. [9] presented a system to classify images using image descriptors generated from general purpose semantic annotations obtained through an external API. They propose using a semantic image descriptor of fixed dimension. Each entry of this descriptor corresponds to a label from a set of predefined labels and contains the probability of that label representing the image.

In [10], a dual 2D and 3D system to solve the search-and-find task is presented. The first 2D-based part of the system classifies wide scenes in order to decide whether to continue exploring in that direction. It uses texture information as the input for a DNN to automatically generate full descriptions from generic indoor images. These descriptions are syntactically and semantically processed in order to help a robot select which rooms to visit for a certain task. For the second part of the system, the 3D-based one, it takes advantage of the geometric information to look for specific objects within the scenes. Making use of 3D segmentation and classification pipelines, it is able to classify the different parts of the scene and locate specific objects. The approach presented in this paper is similar with this work in the sense that semantic relationships among words are used to predict the possible location of objects.

Semantic relationships are also exploited in [11] for object search in large environments. Uncertain environment spatial semantics are built and used in combination with imperfect semantic priors obtained from Internet databases in order to improve the efficiency of the search. This work differs from the one presented here in that it proposes reasoning about unexplored parts of the map, however it is similar to the PLSA in the sense that general semantic knowledge is used and applied in combination with extracted semantic cues to reason about locations of interest.

Regarding generic active object search, the work by Saidi et al. [12] should be pointed out. It models the search task as an optimization problem with the goal of maximizing the target detection probability while minimizing the distance and time to achieve the task. Visibility maps are computed based on a world model in order to search for a local maxima which is supposed to be the new sensor placement in the next step. To evaluate the interest of a given configuration multiple variables are used: the probability of detecting the object, the new volume that will be seen, and the cost in time and energy to reach that configuration. However, in contrast to the PLSA, no visual information is shared from one task to another or from previous experiences.

Another interesting solution exploiting object–object relations for active object search is presented in [13]. The work estimates object–room probabilistic relations that are extracted from a database and matched to defined ontological concepts in order to determine the next room to approach. These relations are used to compute the probability of finding the target object given a set of objects previously seen. This probability is then used to decide which object to approach next in the search process. This is similar to the PLSA in the sense that the relative location of objects to each other is exploited in order to optimize the search. However, the solution presented here differs in that it makes use of the semantic relationship among labels instead of probabilities as a measure of distance between objects.

Object search and localization of objects in indoor environments dealing with low resolution images is also addressed in [14]. The visual search mechanism is based on a combination of receptive field cocurrence histograms and the object recognition through SIFT features [15], while the view planning strategy takes into account the layout of the environment and the specific constraints of the object. They also improve concurrent multiple object search through an optimized use of their shared camera zooming. This work heavily relies on SIFT features, which are specific for object instances and cannot be directly used to detect general classes of objects. This could be extended with new state-of-the-art object recognition techniques (e.g., DNN-based solutions) to improve the accuracy and reduce the steps of the process.

3. Passive Learning Sensor Architecture

The Passive Learning Sensor Architecture is designed to work in indoor environments, passively acquiring relatively low resolution images where the objects to detect are seen from a far distance. It detects possible object locations through multimodal information, combining semantic language data with image information. The architecture consists of a processing pipeline of four main steps, as illustrated in Figure 2. The first step, Cognitive Attention (CA), selects images that show a container with potential objects in it. From these images, it extracts Regions of Interest (ROI), i.e., square parts of the image, where an object container is seen. The second one, called Cognitive Subtraction (CS), performs a segmentation of this ROI using the internal world model. This process extracts, within the regions selected during the first step, those regions potentially corresponding to known or unknown objects lying in the object containers. Afterwards, a Convolutional Neural Network (CNN) step sets labels according to the possible classes (i.e., object types) for the image regions obtained in the previous step. Finally, a Semantic Processing (SP) step uses a learned semantic model to improve the labels from the CNN and maximize the probability of finding the correct container for the object searched. After processing the images, it produces average semantic descriptors for each container found that will help in the later object search task. This section details the different stages of the pipeline of the sensor architecture. We use the case of a social robot looking for a mug to better explain the work flow of the architecture.

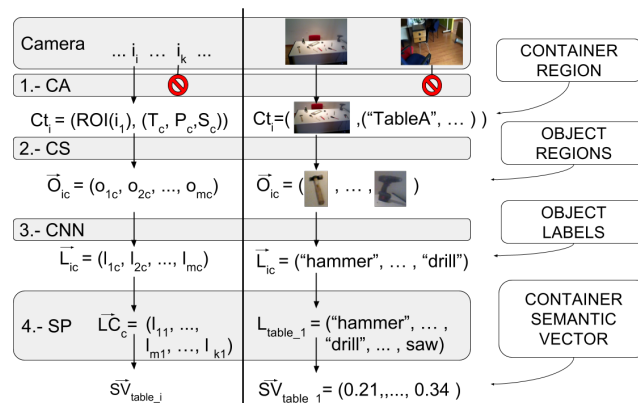


Figure 2. The four main steps of our passive learning architecture. The left-hand side of the vertical black line describes the output of each step in a mathematical notation, while the right-hand side shows it visually. Explanations on the outputs are given on the outer right descriptions. The forbidden sign means the image will be discarded.

3.1. Cognitive Attention

This first stage of the architecture filters the images where an object container is seen, it detects regions of interest corresponding to object containers, and provides such regions along the information of the container in the image (identifier and geometrical properties) to the next stage of the pipeline. For each image, I_i accepted; this first stage provides the next stage with a tuple Ct_i containing the region of interest of the image that shows the container $ROI(I_i)$, along with that container's specific information such as its type (T), pose (P), and shape (S). See Equation (1).

$$CA(i_i) = Ct_i = (ROI(I_i), (T, P, S)) \quad (1)$$

In order to decide if a container is seen (i.e., lies in the frustum of the camera), the architecture must have access to an estimation of the pose and shape of the containers. Given this information and the parameters of the camera, the first step is to check if any of the known object containers are seen in the current image and to estimate the area of the image occupied by it. This process of making the sensor focus its attention on the parts of the image in which containers can be seen is called Cognitive Attention (CA). Thanks to this step, the architecture only takes into account those pictures taken when a container lies within the frustum of the camera, and only processes the regions of the images showing those containers (in 1.-CA from Figure 2 you can see images going through and images not going through). This helps reduce the probability of false positives coming from parts of the environment that the architecture is not supposed to consider.

The cognitive model used in this implementation of the PLSA is Active Grammar-based Modeling (AGM) [16]. AGM cognitive models are multi-graphs where nodes and edges are typed and can be attributed with metric properties. The type of a node is used to denote the kind of entity it represents, whereas the type of an edge denotes the kind of relationship among the linked symbols. Metric properties can also be included in the nodes and in special edges designed to this end. The metric properties of nodes and edges allow to automatically generate a geometric model of the environment from the cognitive model. Additionally, these geometric models can be easily used to perform collision checks, to estimate relative poses and camera projections, and other operations. An AGM world model that contains the robot, rooms and tables is shown in Figure 3. However, other alternatives to this cognitive model can also be used (e.g., [17,18]) as long as they allow storing and accessing the necessary information, and support the computation of the contour of the containers in the images. Given the cognitive model, once an image is considered to have a container, the contour of that container is projected from 3D to the 2D space of the camera image. If at least 80% of the of the container lies within the image, then that region is selected and the next stage of the PLSA is triggered, otherwise the image is ignored.

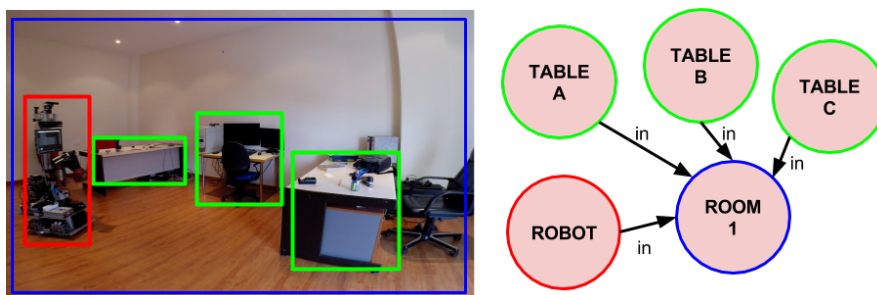


Figure 3. (left) A social robot uses the PLSA to search for objects in an apartment; (right) A schematic view of the AGM cognitive model, with the symbols of the robot, one room and three tables.

Therefore, the CA stage selects the images from the camera that show containers and outputs the tuples corresponding to the detected areas of those containers along with the container information from the cognitive model. The output of the cognitive attention stage $CA(I_i)$ is a vector of tuples Ct_i , where Ct_i tuples are composed of the region of interest and the container information (i.e., its type, pose and size), as can be seen in Figure 2, 1.-CA step.

For the case of a social robot looking for a cup in the apartment where all containers are tables, this stage selects the images containing the tables as it moves around the apartment. It uses the information from its own world model to detect the location of the robot and the tables at any moment. From those images, this step selects the ROIs containing the tables and passes them along, with the shape and pose of the corresponding object container, to the next step.

3.2. Cognitive Subtraction

The second stage of the pipeline performs an additional segmentation step called Cognitive Subtraction (CS). It takes as input the tuples of the regions of interest obtained by the previous stage of the pipeline along with the container information, Ct_i , and generates a series of sub-regions of interest out of each container image, corresponding to possible objects o_{ic} . These sub-regions, which are expected to contain a single object each, are associated to its container and constitute the output of this stage, e.g., for one image, associated with a container c , it will produce a set of sub-images $\vec{O}_{ic} = (o_{1c}, o_{2c}, \dots, o_{mc})$ (see 2.-CS in Figure 2):

$$CS(Ct_i) = \vec{O}_{ic} = \{o_{1c}, o_{2c}, \dots, o_{mc}\} \quad (2)$$

Following the general idea suggested by Cotterill et al. [19], among many others, that rational behavior is “internally simulated interaction with the environment”, CS uses the cognitive understanding of the environment provided by the previous step in order to perform a proper segmentation of objects on the container. Thereafter, the CS algorithm uses the information about the object container and detects differences between the data acquired from the sensors and synthetic data obtained by *imagining* the output of the sensors, given the current world model.

The information obtained from the previous step contains the type, pose and size of the containers. The CS step triggers the specific algorithms to perform the subtraction of known elements in order to detect the new unknown ones. The following are the steps taken for the specific case of tables as containers in order to subtract the unknown data (unknown objects on the tables) from the real world data:

1. Random sample consensus [20] is used to estimate the plane of the table using the point cloud of the scene acquired with the RGB-D camera. The border of the table is estimated using a convex hull of the 3D points laying within the table plane. Using this border information, an imaginary prism is created on top of the table. All the 3D points inside this prism are extracted and considered to belong to the unknown objects lying on the table.
2. Different object point clouds are segmented using euclidean distance clustering [21]. A threshold distance to determine if points belong to the same object or to a new one is used (for the experiments conducted, 0.01 m).
3. Candidate object point clouds are transformed to image coordinates and the image region corresponding to the object candidate is segmented.

Figure 4 illustrates a sample of the results obtained after these steps. The regions of interest, marked in the figure, correspond to parts of the environment that are unknown to the robot and are probably objects lying on the table. Therefore, they are fed to the next step for labeling purposes.

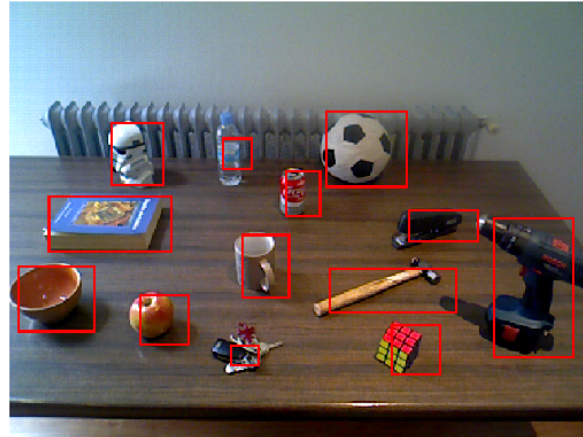


Figure 4. Example of segmentation by the cognitive subtraction algorithm. It must be taken into account that all the cognitive steps are made over the 3D data and projected back to the image.

In the example of the social robot looking for a cup, the output of this step will be the segmentation of the objects lying on top of the table. Using the table selected along with the cognitive information from the previous step segmentation, the robot performs the specific steps to segment the objects on top of the table. The segmented parts, passed on to the next step, are the object candidates to be labeled.

3.3. CNN Classification Step

The third step of the PLSA classifies the image regions obtained from the previous step. It produces a label l_{ic} for each of the object candidates regions o_{ic} obtained in its input (see Equation (3) and 3.-CNN in Figure 2).

$$CNN(\vec{O}_{ic}) = \vec{L}_{ic} = (l_{1c}, l_{2c}, \dots, l_{mc}) \quad (3)$$

Any algorithm able to perform this task can be used for this step. This processing of the object candidate regions is open to the usage of new algorithms that might, in the future, improve the current object classification state-of-the-art.

The current implementation uses a very deep Convolutional Neural Network (CNN) based on deep residual learning [22]. The main characteristic of deep residual learning is the nature of their building blocks, illustrated in Figure 5. In the figure, $F(x)$ represents the residual learning function, where x is the input of the layer N that gets added to the residual value at the layer $N + 1$. This essentially drives the new layer to learn something different from what the input has already encoded. This is repeated for all the layers in the network.

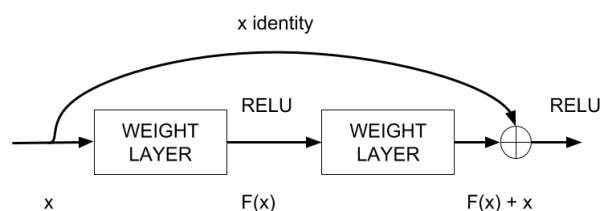


Figure 5. A building block of the residual learning process in our DNN.

The other advantage of this setup is that such connections help to handle the vanishing gradient problem in very deep networks, which slows down the training of front layers. This is defined formally in Equation (4); being x and y the input and output vectors respectively and having $F(x, W_i)$ as the residual mapping to be learned. For the example in Figure 5 that has two layers, $F = W_2\sigma(W_1x)$ in which σ denotes the rectifier linear unit (RELU) [23].

$$y = F(x, W_i) + x \tag{4}$$

Figure 6 shows the whole architecture of the 152 layers CNN with residual learning. At the top part of the figure, the network configuration is described while at the bottom it shows the graphical setup of the layers. Although the network used here is much deeper than others, it has a lower complexity (measured in FLOPs). While in a traditional approach, a layer has to generate a whole desired output, thanks to the design of the building blocks of the residual networks, their layers are only responsible for, in effect, fine tuning the output from a previous layer by just adding a learned residual $F(x)$ to the input x . It uses 3-layer bottleneck blocks for which each residual function F uses a stack of three layers instead of two. The three layers are 1×1 , 3×3 , and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions.

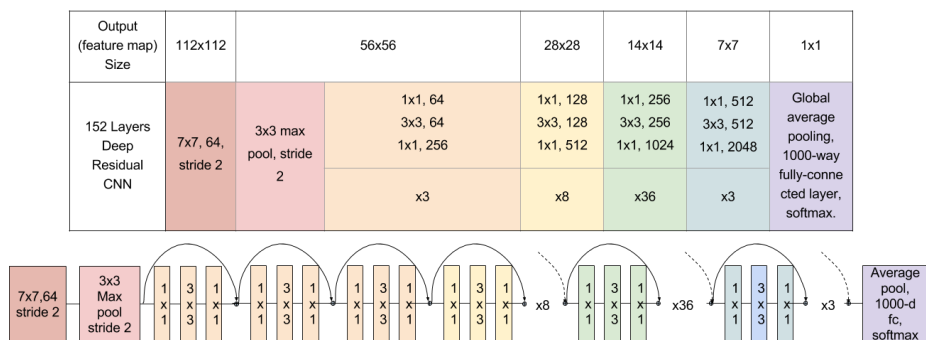


Figure 6. Architecture of the 152 layers CNN with residual learning.

Most of the time, training these networks is a heavy task in terms of time and hardware prerequisites. Therefore, the model of the CNN used in the experiments was trained using the generic ImageNet dataset. Also, this configuration proved to perform better than others tested here, as deeply explained in Section 4.

In the use case of the social robot, this step would only take the small images produced in the previous step, corresponding to objects on the table, and apply a label to each one of them. These labels, associated to their container, are then passed on to the next semantic processing step.

3.4. Semantic Processing

The Semantic Processing step (SP) takes all the labels produced by the previous step L_{ic} for all the images and groups them according to their own containers c . Therefore, a vector of all labels from all images for each of the seen containers is processed separately $LC = L_{1c} + L_{2c} \dots L_{pc} = \{l_{c1}, l_{c2}, \dots, l_{cm}, \dots, l_{ck}\}$. As a result, an average semantic vector \vec{S}_c is produced for each container c (see step 4.-SP in Figure 2).

Since the output of this step is a common vector for each container, all objects in the container will be considered when looking for a certain item. This process takes advantage of the fact that, in household environments, objects located next to each other are usually semantically related in some way or another. Sometimes, some mixing can be found but in general it can be said that they are usually grouped in spaces (toys, kitchen utensils, office supplies, etc.). As we use average semantic vectors to decide if the object is

on a certain container, if an object is “out of place”, it might not be found on the first container selected. In the use case of the social robot, when looking for the cup, even if it is misplaced, the robot will go look for it in the kitchen, not around the tools or any other place, similarly to what a human might have done.

For this step, the skip-gram model [24] is used, also commonly known as *word2vec*. It is a “sallow” word embedding model which learns to map discrete words, represented by an id, into a low-dimensional continuous vector space using the distributional properties of the word observed along a raw text corpus (a large and structured set of texts). The word vectors learned can be used for different research purposes. One of the most common ones (the one used here) is the computation of the distance between word vector representations as a measure of word semantic similarity as, commonly, in the corpus provided in the training, similar words appear close to each other. The model used in this step for the current implementation of the PLSA is also a generic one. It has been trained on texts obtained from the Google News dataset (with more than 100 billion words).

This last step is intended to expand and improve the image labeling results by using the semantic relationships between the labels obtained on a certain container. Due to the low resolution of the images provided to the CNN step, the results obtained after it are still not robust enough for real applications. Additionally, the label used to identify the objects to search are often just a synonym or a similar word of the one the CNN is using, resulting in never finding the required object (e.g., the system provides the label *mug* to an object and the user is looking for a *cup*). This is also a case where semantics can help find a certain object (e.g., no *cup* has been detected but the detection of *coffee* guides the PLSA to the same container).

Vector representations help expand the semantics of the labels assigned to each container. Therefore, for every set of labels obtained for a certain container during the CNN step, \vec{L}_{ic} an average Semantic Vector (SV) is computed $S\vec{V}_c$. This way, each container has a single semantic feature vector. These average vectors consider all labels in a certain location as a whole, minimizing the effect of false positives provided by the CNN step in the final object search.

Once the user of the system wants to find an object with label l_o , the semantic vector representation $S\vec{V}_{l_o}$ of the label is computed using the learned skip-gram model. Afterwards, the Semantic Similarity SS to each container c is calculated as the cosine distance (dot product) of the representation of the label $S\vec{V}_{l_o}$ and the semantic vector of the container $S\vec{V}_c$ (see Equation (5)). The higher the value obtained, the better result and the closer, semantically, the object searched is to an object container.

$$SS = S\vec{V}_{l_o} \cdot \frac{1}{n} \sum_i^n \vec{L}_{ic} = S\vec{V}_{l_o} \cdot S\vec{V}_c \quad (5)$$

In the use case of the social robot wondering around the apartment, an average semantic vector representation would have been calculated for each of the tables. Afterwards, when looking for a *cup*, the semantic distance of the semantic vector representation of *cup* against all the vectors from the tables would be calculated. The higher the value of a table the more probable it is to find the object there. This is visually shown in Figure 7, as you can see the table selected to approach would be the one with the kitchen utensils.

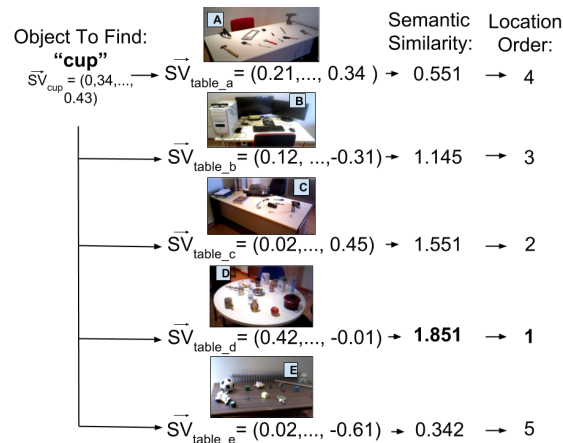


Figure 7. The process of finding an object location by obtaining the semantic similarity of a semantic vector with the known containers.

4. Experiments

To demonstrate the effectiveness of all stages of the PLSA, each of them has been individually tested using a social robot. Additionally, an experiment has been conducted to study how the number of images used to inspect the containers affects the results.

Figure 8 shows the environment in which the experiments took place. Five different object containers (tables in this case) are distributed among two rooms of an apartment, with five different types of objects on them. The disposal of the objects follows the principle that, in indoor environments, those with similar purposes are usually found in the same places. The tables in the apartment are configured for the experiments as follows (labeled as shown in Figure 8): table A contains hardware tools, table B has a computer and other tech gadgets, table C has office supplies, table D has kitchen utensils, table E contains different toys. It is worth noting that some of the objects are often labeled differently depending on the person asked (e.g., some people would label an object as a “toy” while others would call it a “plush”). Therefore, asking the robot about the location of an object using a particular label requires the system to be able to generalize.

The social robot is equipped with an RGB-D camera and operates in the apartment with an implementation of the PLSA. The experiments have an initial phase in which the robot wanders around the apartment, passively taking pictures of the tables, labels the objects lying on them and calculates the average vector representation of each container. Pictures are considered when a table is in the field of view of the camera, at less than 2 m and at a frame rate no higher than 1 Hz. These pictures are then used to train the average semantic vector representation of the tables. After the initial phase, the robot is asked to use the multimodal information (combination of images features and language semantics) to locate 20 objects among the ones on the tables. The results obtained are compared with the following combination of image segmentation and state-of-the-art CNN image recognition systems:

- GoogleNet [25] is a 22 layers deep network (27 if pooling is taken into account) that makes use of “inception modules” which basically act as multiple convolution filter inputs, that are processed on the same source, while pooling at the same time. Another training of this network but without relighting data-augmentation was also tested (GoogleNet2).
- AlexNet, by Krizhevsky et al. [26], consists of eight layers, of which five are convolutional layers, with some of them being followed by maxpooling layers. The other three layers are fully-connected layers with a final 1000-way softmax.

- Very Deep Convolutional Networks by Simonyan et al., presented in [27] (VGG16) consist of a series of thirteen convolutional layers (also with maxpool in between), followed by three fully connected layers.
- Regions with Convolutional Neural Network (R-CNN) [28] performs localization and classification of the objects in the image. It generates category-independent region proposals, then a convolutional network extracts a fixed-length feature vector from each region and finally the third module, which is a set of class-specific linear SVMs, scores each feature vector. Since it performs localization by itself, no previous segmentation step is added to this network.



Figure 8. Setup of the experiment: On the top, the two room of the real setup; on the bottom-left, the visualization of the cognitive model of the setup; on the bottom-right, the tables and objects on them: table A contains tools, table B tech gadgets, table C office supplies, table D kitchen utensils and table E toys.

These networks competed in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [29] and the PASCAL VOC [30] competitions and obtained top positions in their rankings. In the experiments presented in this section, they were tested with both, generic and specific training data.

The following experiments are as follows. Section 4.1 tests how the number of images processed per container affects the whole system. This is done to look for, if it exists, a maximum number of images after the PLSA should stop processing for a certain container. For this, we watch the results of the whole PLSA as we add images to a certain container. Afterwards, the Cognitive Attention step is tested by comparing the results of the PLSA with this stage and without it, in Section 4.2. In Section 4.3, the output after the the first three layers and the whole architecture is compared with combinations of previously mentioned state-of-the-art algorithms. This proves not only that the results of the three first layers improve state-of-the-art algorithms but also that they improve when adding the final Semantic Processing step. Finally, Section 4.4 shows how specific retrains of these networks do not actually improve the results of the PLSA, which justifies the use of generic training data sets for the architecture models.

4.1. Tests on Image Buffering

The purpose of this test is to study the evolution of the Semantic Similarity \mathbb{S} of an object to the containers as the size of the PLSA image buffer is increased. The number of labels produced and taken into the semantic step increases as more images are added to be processed. This can make the final results vary. To test this, results were obtained starting from a buffer size of one image, and the size

was increased one by one up to a buffer size of 43 images. Figure 9 shows how the semantic similarity of the labels of objects with table A evolves as images of such a container are added to the buffer. In Figure 9a, only objects present on the table are tested against the labels from table A. As a general rule, it can be said that the similarity with labels of present objects keeps increasing or stabilizes as images are added. The contrary occurs in Figure 9b, where objects not present on the table are queried for table A. In this last case, similarity decreases as images are added. Therefore, it can be concluded that adding images to the buffer either improves or it does not negatively affect the results of the architecture. Little improvement is appreciated for a buffer size higher than 20 images. Because of these results, during the rest of the experiments, the buffer size was set to store the information of the last 30 images for each object container.

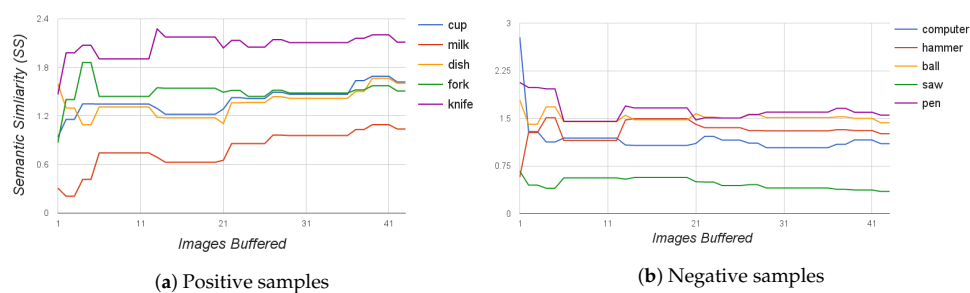


Figure 9. These graphs show how the semantic similarity between objects and the SV of table A evolves as more images of table A are added to the buffer of the PLSA. The higher the Semantic Similarity value is, the higher the likelihood of the presence of the object. (a) objects queried are present in the table; (b) objects queried are not present in the table.

4.2. Cognitive Attention Tests

To test the Cognitive Attention stage, this experiment compares the results obtained with and without this stage. The setup of the experiment is depicted in Figure 10. Table 1 shows the results obtained. Using the Cognitive Attention step, only the table region of the image is considered, while without it, the full image is used. Results prove the effectiveness of this step as the PLSA obtains a higher success rate when using CA.

Additionally, to prove that CA is not only beneficial to the PLSA, this step was also tested with another segmentation and labeling technique: the Top-Hat (TH) [31] combined with three of the main DNN presented at the beginning of Section 4. As can be appreciated in the table, the results are equal or better when focusing on the selected table region of the image.

Table 1. Success rate of the PLSA and other algorithms' configurations when using full images of the ROI obtained from the Cognitive Attention step.

Method	Full Image	Cognitive Attention
PLSA (no semantics)	0.4	0.65
TH + GoogLeNet	0.35	0.35
TH + GoogLeNet2	0.35	0.35
TH + AlexNet	0.2	0.35
TH + VGG16	0.55	0.6

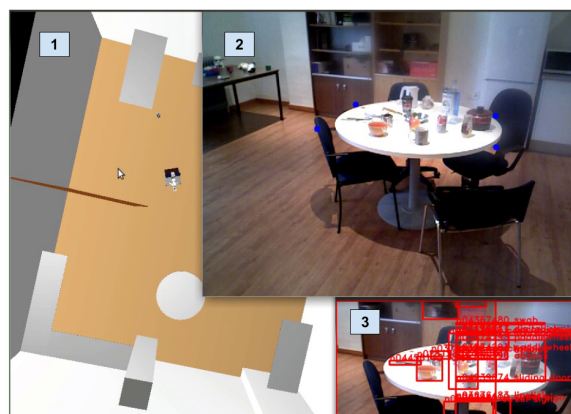


Figure 10. Cognitive attention in practice: 1. Shows the visualization of the cognitive model; 2. Is the real Primesense sensor camera with augmented reality, projecting the cognitive location of the table marked with four blue dots; 3. Shows the specific part of the image to process along with a Top-Hat based segmentation and a GoogLeNet classification.

4.3. Tests with Networks with Generic ImageNet Training

The full PLSA was compared with a combination of state-of-the-art object detection and recognition systems. For this, three main segmentation algorithms were used:

- Top-Hat [31]: A morphology transformation based algorithm commonly used for segmentation purposes.
- Multiscale Combinatorial Grouping [32]: An algorithm for bottom-up hierarchical image segmentation.
- The Cognitive Subtraction: Algorithm explained in Section 3 and stage two of the PLSA.

These segmentation algorithms were combined with the CNN architectures with best results in the top challenges in computer vision, addressed in Section 4. To test the system, it was queried with 20 objects and asked to locate them among the five object containers considered for the experiment (tables *A*, *B*, *C*, *D* and *E*). If the first choice of the algorithm was the correct table, it was counted as a success, otherwise it was considered a failure.

The first three steps of the PLSA are compared with a mix of the segmentation algorithms combined with the DNNs explained configurations with generic trainings. Table 2 shows the normalized success rate of all of them. For results on the second column (Direct Match), a direct naive match, in which labels of objects to find are directly matched with the labels detected in the obtained images, is used. The third column shows the results when adding the semantic processing step. The results prove how the PLSA outperforms all of the other solutions tested and, at the same time, how a semantic processing step brings improvements not only to PLSA but also, in most of the cases, to the labels resulting from the rest of the CNNs. All the DNN algorithms used for the results in Table 2 use models trained with the generic ImageNet database.

Table 2. Success rate of the object search test when using the first three steps of the PLSA against top CNN algorithms and results when adding the SP step.

Method	Direct Match	Semantic Processing
PLSA	0.65	0.75
TH + GoogleNet	0.35	0.45
TH + GoogleNet2	0.35	0.45
TH + AlexNet	0.35	0.1
TH + VGG16	0.6	0.6
MCG + GoogleNet	0.5	0.5
MCG + AlexNet	0.15	0.25
MCG + ResNet	0.55	0.55
MCG + VGG16	0.55	0.45
CS + GoogleNet	0.45	0.5
CS + GoogleNet2	0.6	0.65
CS + AlexNet	0.2	0.25
CS + VGG16	0.45	0.45
R-CNN	0.4	0.0

4.4. Tests with Networks with Fine-Tuned Training Datasets

These experiments perform the same tests as the previous one, but using fine-tuned data sets. Instead of using the models trained with the generic ImageNet database, a specific fine-tuned retrain on a reduced set of classes was used. For these tests, the following changes were made to the models of the DNNs:

- GoogLeNet2_ft: this model is the ImageNet trained GoogLeNet2 model with a retrain on the last full-connected layer for 138 classes.
- VGG16_fc1: is VGG16 fine-tuned on 1000 categories by simply training on new images.
- VGG16_fc2: uses the VGG16 model but retraining the last fully-connected layers on 136 categories.
- VGG16_fc3: is VGG16 fine-tuned on 44 categories by changing the last fully-connected layer.
- R-CNN_m: is using the pretrained R-CNN with bounding boxes (region proposals) given by MCG instead of Selective Search.

As well as in Table 2, the first three steps of the PLSA are compared to different combinations of segmentation algorithms and DNNs, and then the Semantic Processing step is added to all the combinations to test its effectiveness. The resulting success rates are shown in Table 3. They prove how the PLSA (even with a generic training) still outperforms the success rates of fine-tuned networks. However, it is worth noting that the semantic processing step does not perform as good as on the previous tests with these training configurations.

In general, the success rates of the direct match approach (without the Semantic Processing step) show better results for these retrained networks than for the generic ones. However, when adding the semantic processing step, these success rates do not seem to improve, they even worsen in some cases. In order to further investigate why this step does not improve the networks with fine-tuned models, some extra experiments were performed. The average semantic similarity of the labels applied to each container were computed. Average values of the results of models trained with the generic database and the retrained models were compared. Results of these computed average semantic similarity are shown in Table 4. For each table, an average semantic distance among all the vector representations of the labels found for that table was computed. Since the semantic similarity is an inverse distance measure, a higher value means the label representations of a table are “semantically closer” in the search space. For all the cases, the average similarity among all the labels applied to tables in the generic trained networks is higher than the similarity for the same networks when retrained with a lower amount of classes. These results mean that labels applied to containers by the generic training are closer to each other in the search space than the ones from the fine-tuned trainings, which means they are also semantically closer.

Table 3. Average success rate of the object search test with the PLSA and fine-tuned networks. The second column shows the results of not including Semantic Processing and the third one shows the results when including this step.

Method	Direct Match	Semantic Processing
PLSA	0.65	0.75
TH + VGG16_fc1	0.5	0.4
TH + VGG16_fc2	0.65	0.45
TH + VGG16_fc3	0.4	0.4
TH + GoogleNet2_ft	0.35	0.6
MCG + VGG16_fc1	0.5	0.4
MCG + VGG16_fc2	0.55	0.55
MCG + VGG16_fc3	0.55	0.4
MCG + GoogleNet2_ft	0.35	0.4
CS + VGG16_fc1	0.65	0.6
CS + VGG16_fc2	0.6	0.5
CS + VGG16_fc3	0.5	0.5
CS + GoogleNet2_ft	0.45	0.4
R-CNN_m	0.35	0.2

Table 4. Average word2vec semantic similarity between labels detected per table. Rows in bold correspond to the original models without fine tuning.

Method	Table A	Table B	Table C	Table D	Table E
MCG + VGG16	0.6014	0.3906	0.37714	0.33766	0.4097
MCG + VGG16_fc1	0.19741	0.23967	0.19976	0.21568	0.28264
MCG + VGG16_fc2	0.18415	0.22191	0.18938	0.17405	0.24179
MCG + VGG16_fc3	0.2178	0.19943	0.18505	0.20464	0.23815
MCG + GoogLeNet2	0.28346	0.2965	0.24557	0.2136	0.36935
MCG + GoogLeNet2_ft	0.18915	0.19017	0.18314	0.18309	0.32282

For a better visualization of this issue, t-SNE [33] was used. It is a dimensionality reduction technique well suited for the visualization of high-dimensional datasets. Using this technique, Figure 11 shows a plot of the vector representations of labels obtained for table A, when using a generic training (Figure 11a) and when using a fine-tuned one (Figure 11b). Since generic trainings have more classes, sometimes they generate *light fails*. A *light fail* is a name used to denote failure made with a word that is not very far in meaning to the correct one (e.g., mug instead of cup). This light fails to help maintain the \vec{S} of the container around the similar space, almost as if it were a correct answer, therefore not considerably affecting the final result. However, when using the models retrained with a reduced set of classes, and probably influenced by the lack of vocabulary available in their results, the fails are usually not so *light*. This means that a fail, more often than with the big ImageNet generic trained models, will be a totally different word (e.g., *screwdriver* instead of *cup*). This makes the location in the search space of the \vec{S} of a container very affected by the fails of these models, which makes it have a lower semantic similarity (a higher distance to the words in the search space) to the actual correct labels of the objects it contains, lowering its general success rate. Therefore, it can be concluded that generic training datasets with a larger number of classes are more likely to benefit from the semantic processing step of the PLSA.

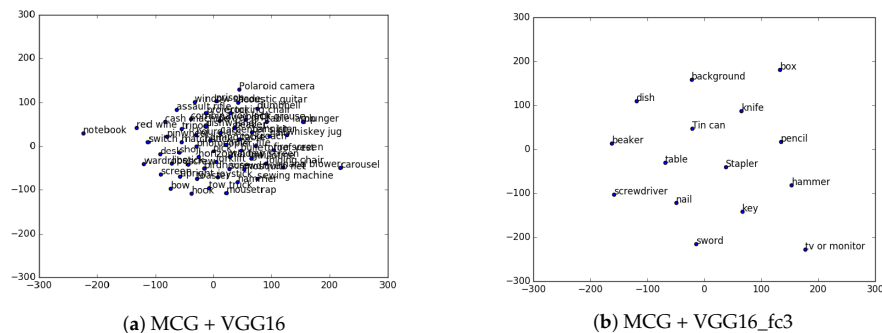


Figure 11. Visualization, using t-SNE for dimensionality reduction, of labels applied to table A.

5. Conclusions and Future Work

The PLSA was introduced and deeply described. The experimental results obtained from its use in a household social robot were also presented. It was demonstrated that it outperforms state-of-the-art algorithms both with generic and fine-tuned training datasets. The different steps of the architecture were deeply tested, showing how they contribute to the whole architecture and to its final performance. It is believed that this solution constitutes a firm candidate to be used as a first step to guessing object locations either in social robotics tasks or any other solution that involves search functions in broad scenarios.

Further work improving the different steps in the architecture can be made. Specially, it would be interesting to explore new alternatives for the semantic processing step. Other word semantic relationships can be tested in an effort to improve this process. It can also help discover what specific training setups can actually benefit from it. Different strategies for selecting specific labels that might become representative of certain locations might help improve results.

The current output of the system only handles one query at a time and this may also be extended. Taking into account multiple object searches at the same time might help reduce the time of high level tasks with a strong dependence on the find-and-search process, e.g., having a robot clean an apartment or cook something in the kitchen. For these tasks with multiple objects, instead of one container, a best path to find several objects should be given as an output. Therefore, the distance between containers should also be taken into account.

This architecture is meant as a first guess on where to look for an object. Further work might be required to strengthen final results and have actuators that actually verify the existence of the object or even robots that fetch and bring the item to a specific location. Current implementation of the PLSA provides a primary guess in the search task; in order to fully integrate the architecture into higher level plans, more search and verification steps will be needed to assure a final successful execution of the plan. For example, when two objects with the same label are found, an extra step could be added in this verification step to decide which one to choose.

Supplementary Materials: A video of the robot setup and run can be found at: <https://youtu.be/7dWlc5zc8vo>. All the code involved in the experiments are part of the RoboComp framework which is freely available at: <https://github.com/robocomp>.

Acknowledgments: This work has been partially supported by the MICINN Project TIN2015-65686-C5-5-R, by the Extremaduran Government project GR15120, by MEC project PHBP14/00083. The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of GPU hardware used for this research.

Author Contributions: Marco A. Gutiérrez and Luis J. Manso designed and implemented the architecture. They also run different tests and wrote part of the paper; Harit Pandya implemented and trained the deep neural networks used on the third stage. He also run several tests that helped improve the architecture; Pedro Núñez helped with the analysis of the data and paper writing.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AGM	Active Grammar Model
API	Application Programming Interface
CA	Cognitive Attention
CNN	Convolutional Neural Network
CS	Cognitive Subtraction
DNN	Deep Neural Network
FLOP	Floating Point Operations
GPU	Graphic Process Unit
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
MCG	Multiscale Combinatorial Grouping
PLSA	Passive Learning Sensor Architecture
R-CNN	Regions with Convolutional Neural Networks
RELU	Rectifier Linear Unit
RGB-D	Red Green Blue Depth
ROI	Region Of Interest
SIFT	Scale Invariant Feature Transform
SP	Semantic Processing step
SV	Semantic Vector
SVM	Support Vector Machine
TH	Top-Hat
t-SNE	t-distributed Stochastic Neighbor Embedding
VGG	Visual Geometry Group
VOC	Visual Object Challenge

References

1. Kita, Y.; Kanehiro, F.; Ueshiba, T.; Kita, N. Strategy for Folding Clothing on the Basis of Deformable Models. In Proceedings of the 11th International Conference on Image Analysis and Recognition (ICIAR 2014), Vilamoura, Portugal, 22–24 October 2014; Campilho, A., Kamel, M., Eds.; Springer: Cham, Switzerland, 2014; Part II, pp. 442–452.
2. Doty, K.L.; Harrison, R.R. Sweep Strategies for a Sensory-Driven, Behavior-Based Vacuum Cleaning Agent. In Proceedings of the AAAI 1993 Fall Symposium Series, Raleigh, NC, USA, 22–24 October 1993; pp. 1–6.
3. Bollini, M.; Tellex, S.; Thompson, T.; Roy, N.; Rus, D. Interpreting and Executing Recipes with a Cooking Robot. In Proceedings of the 13th International Symposium on Experimental Robotics, Quebec City, QC, Canada, 18–21 June 2012; Desai, P.J., Dudek, G., Khatib, O., Kumar, V., Eds.; Springer: Heidelberg, Germany, 2013; pp. 481–495.
4. Khosravi, P.; Ghapanchi, A.H. Investigating the effectiveness of technologies applied to assist seniors: A systematic literature review. *Int. J. Med. Inform.* **2016**, *85*, 17–26.
5. Kidd, C.D.; Orr, R.; Abowd, G.D.; Atkeson, C.G.; Essa, I.A.; MacIntyre, B.; Mynatt, E.; Starner, T.E.; Newstetter, W. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organizations, and Architecture (CoBuild'99), Pittsburgh, PA, USA, 1–2 October 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 191–198.
6. Szegedy, C.; Toshev, A.; Erhan, D. Deep neural networks for object detection. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 2553–2561.
7. Manso, L.J. Perception as Stochastic Grammar-Based Sampling on Dynamic Graph Spaces. Ph.D. Thesis, University of Extremadura, Badajoz, Spain, 2013.

8. Woodman, G.F.; Chun, M.M. The role of working memory and long-term memory in visual search. *Vis. Cognit.* **2006**, *14*, 808–830.
9. Rangel, J.C.; Cazorla, M.; García-Varea, I.; Martínez-Gómez, J.; Éliasa, F.; Sebban, M. Scene classification based on semantic labeling. *Adv. Robot.* **2016**, *30*, 758–769.
10. Gutierrez, M.A.; Banchs, R.E.; D'Haro, L.F. Perceptive Parallel Processes Coordinating Geometry and Texture. In Proceedings of the Workshop on Multimodal Semantics for Robotics Systems (MuSRoS) and International Conference on Intelligent Robots and Systems, Hamburg, Germany, 28 September–2 October 2015.
11. Aydemir, A.; Pronobis, A.; Göbelbecker, M.; Jensfelt, P. Active Visual Object Search in Unknown Environments Using Uncertain Semantics. *IEEE Trans. Robot.* **2013**, *29*, 986–1002.
12. Saidi, F.; Stasse, O.; Yokoi, K.; Kanehirot, F. Online object search with a humanoid robot. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 1677–1682.
13. Elfring, J.; Jansen, S.; van de Molengraft, R.; Steinbuch, M., Active Object Search Exploiting Probabilistic Object–Object Relations. In *RoboCup 2013: Robot World Cup XVII*; Behnke, S., Veloso, M., Visser, A., Xiong, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 13–24.
14. Sjö, K.; López, D.G.; Paul, C.; Jensfelt, P.; Kragic, D. Object search and localization for an indoor mobile robot. *CIT J. Comput. Inf. Technol.* **2009**, *17*, 67–80.
15. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110.
16. Manso, L.J.; Calderita, L.V.; Bustos, P.; Bandera, A. Use and Advances in the Active Grammar-based Modeling Architecture. In Proceedings of the International Workshop on Physical Agents 2016, Malaga, Spain, 16–17 June 2016; pp. 31–36.
17. Milliez, G.; Warnier, M.; Clodic, A.; Alami, R. A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management. In Proceedings of the 23rd IEEE International Symposium on Robot and Human Interactive Communication, Edinburgh, UK, 25–29 August 2014; pp. 1103–1109.
18. Foote, T. tf: The transform library. In Proceedings of the 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, USA, 22–23 April 2013; pp. 1–6.
19. Cotterill, R.M. Cooperation of the basal ganglia, cerebellum, sensory cerebrum and hippocampus: Possible implications for cognition, consciousness, intelligence and creativity. *Prog. Neurobiol.* **2001**, *64*, 1–33.
20. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395.
21. Jain, A.K.; Murty, M.N.; Flynn, P.J. Data Clustering: A Review. *ACM Comput. Surv.* **1999**, *31*, 264–323.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *arXiv* **2015**, arXiv:1512.03385.
23. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
24. Mikolov, T.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013.
25. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
26. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
27. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
28. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 142–158.
29. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252.
30. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338.

31. Meyer, F. Contrast feature extraction. In *Analyse Quantitative des Microstructures en Sciences des Matériaux, Biologie et Médecine*; Cherman, J.L., Ed.; Rieder: Stuttgart, Germany, 1977; p. 374.
32. Pont-Tuset, J.; Arbelaez, P.; Barron, J.; Marques, F.; Malik, J. Multiscale Combinatorial Grouping for Image Segmentation and Object Proposal Generation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 128–140.
33. Van der Maaten, L.; Hinton, G. Visualizing non-metric similarities in multiple maps. *Mach. Learn.* **2012**, *87*, 33–55.



© 2017 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).

Chapter 5

Semantic Exp. of Auto-Gen. Scene Desc. to Solve Robotic Tasks

Semantic Expansion of Auto-Generated Scene Descriptions to Solve Robotic Tasks

Marco A. Gutiérrez
RoboLab, University of Extremadura, Cáceres, Spain
Email: marcog@unex.es

Rafael E. Banchs
HLT Dept., I2R, A*STAR, Singapore
Email: rembanchs@i2r.a-star.edu.sg

Abstract—When a robot is facing object description based tasks, such as “bring me something to drink water”, it has to semantically relate the concepts on the task with the objects it is able to find. This work expands the semantic scope of words in automatically generated scene descriptions and a given task in order to find a proper match for the robot task. An encoder-decoder pipeline that unifies joint image-text embedding models with multimodal neural language models is used to generate scene descriptions. Then the semantics of those descriptions are extended through word vectors. We improve our previous work by expanding the dimension of the object description by adding the option of negating characteristics of the searched object. Finally we show that we are able to find objects that are in the scene and where not directly referred in the task or labeled by the robot using different words.

Index Terms—object search, semantics, deep neural networks, robotics vision

I. INTRODUCTION

Object searching tasks for robots in unknown environments remains a challenge for the robotics research community. Being able to make a robot successfully perform full pick and place tasks is one of the main objectives of many roboticists. One of the key parts of it is the ability of the robot to properly match the information regarding the object to find with the objects found in the scenes surrounding it. Properly performing the match between the information and the objects found can help robots perform these tasks in a more natural way.

Numerous ways exist for object detection and recognition on scene images. Deep neural network based image labeling and, lately, image description generation are on the state of the art for scene images understanding. Big advances have been done in this field as recent works like [1]-[3] prove. These works make use of deep neural networks in order to produce somewhat accurate descriptions of the image scene. However these captions are usually short and, even though in the case they provide accurate descriptions, they do not fully express

all the information that is contained in the scene. On top of that, same things can sometimes be expressed with different words or people may differ on how they call something that shows up on a certain scene, specially since words can have multiple degrees of similarity [4]. For these reasons we can not solely rely on the words on these descriptions as atomic units that give us all the information we need to match a certain object query. In order to extend the information contained on these generated sentences semantic relations between words can be exploited.

There is a wide range of research works in the field of the analysis of words semantics relations. Works such as [5]-[8] are just an example of some of the most common works in this research area. Some approaches exploit manually created ontologies or taxonomies like WordNet [9] or Freebase [10]. As stated in [11], these works are ontologies that are manually created and maintained in order to provide a means for establishing semantic relations between words and because of that sometimes its further development can be very costly. In consequence, only a determined domains have a suitable ontology, limiting the applicability of similarity measures based on one of them. On the other hand word vectors are a good and fast way to capture semantic relations between words [12], specially when trained over big corpus containing a large amount of words. This makes them easily trainable in the needed semantic scope so the info better matches the application. In our design we decided to handle semantic relations between words by measuring the distance among the word vector representation of those words.

The system presented here weights the semantic relations between a description based search task issued to the robot and scene automatically generated descriptions in order to improve the possibilities to find an object matching the user needs. Our system is even able to handle descriptions that include negations, such as “find an animal that does not bark”. First, the task is analyzed using the Natural Language Toolkit (NLTK) [13] in order to select the key words on it and differentiate negative requirements from positive ones. The neural network encoder-decoder pipeline described in [14] is used in order to generate captions that describe scenes

Manuscript received August 17, 2015; revised December 28, 2015.

from images. Then pre-trained word vectors helps finding semantic similarities between words using the skip-gram model described in [12]. A similarity weight is calculated using the cosine distance in the vector space between the selected words from the descriptive task and the ones in the image generated descriptions. Results are sorted by their calculated similarity weight, the best ones would be the ones with the highest similarity value. This process allows the expansion of the semantic domain of the words on the image generated captions. The system is able to find things that are not explicitly noted in the description sentences. Even in the case of querying for something that is not on the image dataset, the output will still be semantically more relevant than a random ordering of the images.

II. SYSTEM DESIGN

The goal of our system is to look for scenes that contain the information that is described in the task the robot is given. It accepts descriptive tasks in the form of “*get me something to drink water*” or even with negative parts like “*bring me an instrument with no strings*”. When the robot receives a task the system semantically analyzes the sentence detecting the main positive words and the negative ones from the description. It obtains the word vector representations of these words and calculates an average of these vectors by weighing appropriately the negative vectors and the positive ones. Also, as shown in Fig. 1, the system contains a multimodal encoder-decoder pipeline that generates the descriptions for the scenes. The system generates five description candidates for each scene. An average of the cosine distances between the vector representations of adjectives and nouns from these descriptions and the vector representing the description is calculated for each of the scenes. Finally the system provides a rank of best matching scenes according to their weight value. The selected scenes contain the objects that are most semantically related to the words on the description contained in the task.

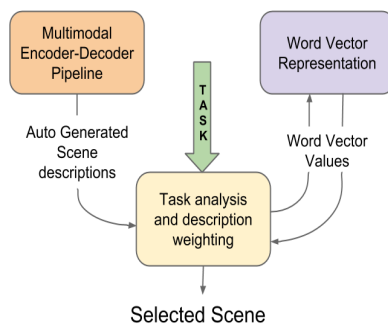


Figure 1. System's architecture

A. Multimodal Encoder-Decoder Pipeline

The system contains an encoder-decoder pipeline that automatically generates descriptions for the scenes. The encoder (see Fig. 2) is learned with a joint image-sentence embedding where sentences are encoded using

long short-term memory (LSTM) recurrent neural networks [15]. Image features from the top layer of a deep convolutional network trained from the ImageNet classification task [16] are projected into the embedding space for the LSTM hidden states. A pairwise ranking loss is minimized in order to learn to rank images and their descriptions.

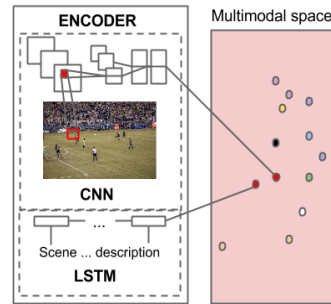


Figure 2. The deep convolutional network (CNN) and long short-term memory recurrent network (LSTM) encoder. It is in charge of learning a joint image-sentence embedding.

As Fig. 3 shows, for decoding, the Structure-Content Neural Language Model (SC-NLM) described in [14] is used, which takes into account the content in the sentences. It is a multiplicative neural language model where the attribute vector is an additive function of the embeddings. These embeddings are conditioned on the embedding vector for the description computed with the LSTM. Allowing the system to make use of large amounts or monolingual text to improve the quality of the language model. Since the embedding vectors share a joint space with the image embeddings, the SC-NLM can also be conditioned on image embeddings after the model has been trained.

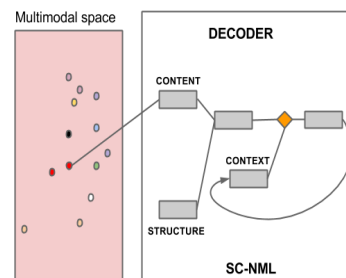


Figure 3. Structure-content neural language model decoder in charge of generating words for the scene description one at a time.

The final output of this pipeline generates are the top five most reliable descriptions for a scene. This is run for each one of the scenes that are in the dataset.

B. Word Semantics Relationships

In order to measure the semantic relationships between words we selected a neural network based tool, since they perform better than Latent Semantic Analysis (LSA) [17] for preserving linear irregularities among words and in terms of computational cost when trained over large

datasets [4], [18]. We use an improved version of the Skip-gram model [12] to find word representations that predict the surrounding words in a corpus. Our system was trained using the negative sampling [19] technique instead of the hierarchical softmax, so it tries to differentiate data from noise by means of logistic regression. Semantic relations on the words of the training data are encoded in a word vector space. The semantic relation between words is measured by the cosine distance between their vector representations. These semantic relationships are used to extend the scene descriptions word meanings when the robot search task is being performed.

C. Word Matching System

This is the module in charge of making the semantic matching and evaluation between the task and the scene descriptions. As the robot receives the task this module analyzes it using NLTK. The positive main words are separated from the negative ones. For this, a basic syntactic analysis of the task is performed along with some basic regular expression matching techniques. Prepositions, pronouns and determinants are ignored as they we do not consider them relevant to the task. Firstly the average vector representing the task description is obtained by summing up the vector representations of the positive words and subtracting the ones from the negative words. This leaves us with a vector representation that will represent the semantic of the description of the task. Then the cosine distance of this vector with the words on the scene description is obtained and an average distance for all of them is finally calculated. This would be the weight of the scene for a specific description in a task, and in consequence a representation of how semantically similar they are.

Finally when all scene weights are computed for the given task they are ranked by their weight value. The output of the system will show the scenes with the highest weight, as those are the ones that are supposed to have a higher semantic similarity with the description on the task. Since they are semantically similar, they should be describing similar things.

III. EXPERIMENTS

In order to perform the experiments the LSTM encoder and SC-NLM decoder of the pipeline described in Section II-A have been trained on sentences from both Flickr30K [20] and Microsoft COCO [21]. We have selected randomly a subset of 1000 images from Flickr30K to use them as a dataset for the scene description generation. These are the ones being considered for a possible selected scene and final match with the task description. The vector space word representation have been trained on a Google News data subset containing about 100 billion words. And the final word vector model contains 300-dimensional vectors for 3 million words and phrases.

Since a manual interpretation of the contents of an image will always be open to criticism of subjectivity [22], there is a high difficulty of quantitatively evaluate

the retrieval effectiveness of our approach. However we will perform a manual evaluation on the output to provide an approximated quantitative evaluation, providing besides the visual output as a support of our experiment results.

For evaluation purposes we have tested the system against a direct word to word matching approach. On this direct matching approach we will select the positive and negative words in the same way we do in our system. Then for the positive words we will add a value of 1 to the overall scene weight if the word in the task description appears on any the scene generated descriptions, otherwise 0 will be added. For the negative words we will subtract 1 if there is a match between the negative word and any word from the scene generated descriptions. The same way as with the positive ones nothing will be subtracted if the word is not found in the descriptions. This measures basically the number of words shared among the description in the task and those from the scene minus the negative words they share. Finally these computed weights would represent the similarity between the scene and the description on the task, the higher the value the more similar they are supposed to be.



a man in a black apron is working on a grill.
a man wearing a black shirt is cooking.
a man with cooking on the ground with his machine.
a young man in a black shirt is cooking on a large grill.
a man is in his left hand.

Figure 4. Top result of the task “find me a barbecue pit”. Note that the words “barbecue pit” do not appear in the generated captions but probably due to the high semantic relation with the word “grill” (0.583 cosine distance) we are able to find it.

For the quantitative results we have evaluated the top five results for six search tasks on both approaches manually giving a score of 1 for correct matches, 0.5 to partially correct matches and 0 to totally wrong matches. We obtained a total score of 25 for our system against a score of 10.5 for the direct match approach, showing the benefits of our semantic expansion approach. We show here a visual excerpt of the obtained results and add some comments on them for a more specific evaluation.¹

Fig. 4 shows the top result, a basic example of the robot process of the task “find me a barbecue pit”. Not

¹Due to space limits we can only show some results here, for a wider overview of all the ones used in the evaluation please refer to: <http://magutierrez.com/description-based-tasks>.

any of the scene generated descriptions show the word “barbecue” among their results. The reason why the system is able to select that picture is due to the high semantic relation between the words “barbecue” and “grill” (cosine distance of 0.583 of their correspondent word vectors representations). In the same way Fig. 5 shows results on different descriptive tasks that had no words for direct matching so the result on the alternative is a total random selection of scenes. However on those pictures our system is still able to provide us with a scene that can match the description from the task. These examples show that even though the description generation system is not reflecting the same words as in the task description we can still match them due to their existing semantic relation represented in the word vector space.

are displayed ordered by similarity score from left to right, keeping the results from our solution on the top row, while the lower one corresponds to the direct match approach output. On the first task (Fig. 5a) our system is able to relate the words “strings” and “instrument” with names of instruments with strings. However on the direct approach only the word instrument is matched from the scene descriptions so even the fact that some guitars are shown as a result is pure coincidence as it could have been any other instrument. On the second task (Fig. 5b) the system gets more confused. However is still better as it can relate the word “drink” with some liquids or drinking situations. Even though, in this case it is not a great result it is still better than a totally unrelated scene such as some of the results on the direct match approach. Some of the errors here are also due to some errors in the automatically generated scene description.

In Fig. 5 we show the results from the two options tested, our system and the direct match approach. Results

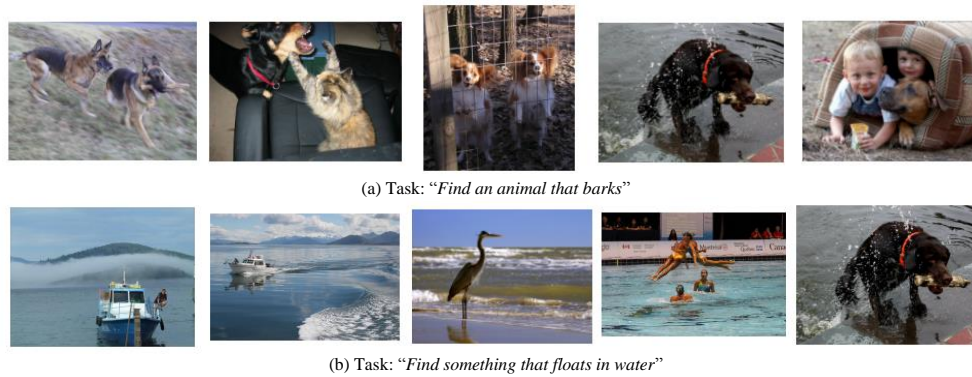
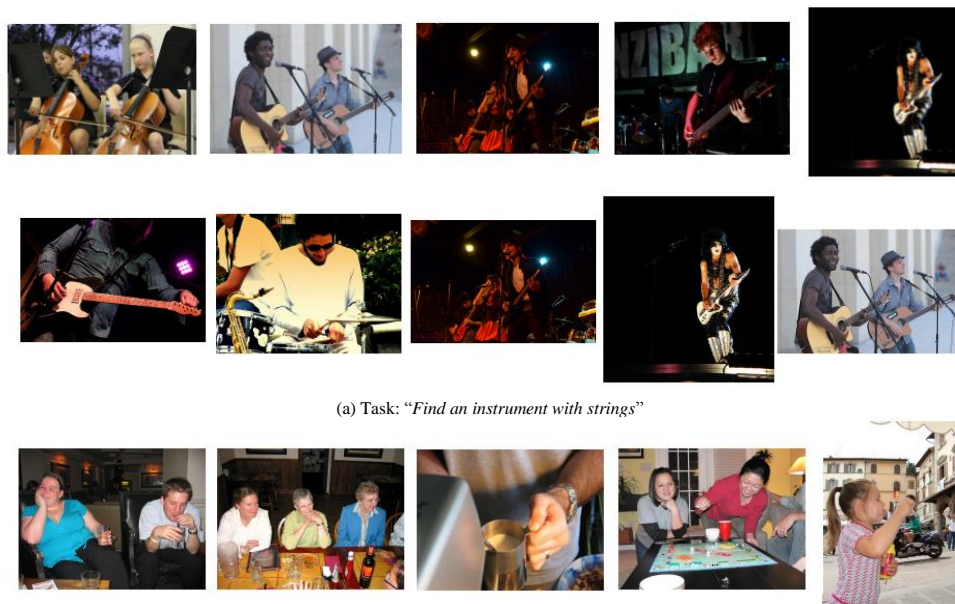


Figure 5. Tasks that had none words in common with any of the scene generated descriptions





(b) Task: "Bring me a drink with alcohol"

Figure 6. First row from left to right of each tasks are our system results compared to the direct match approach on the second row

Fig. 7 uses the novel introduced negation part on the description from the task. This example task is "find a sport with no ball". The first results are good as the system relates some sport with the word ball and its able to discriminate them. It gets some confusion though and there is clearly room for improvement. However we

found out that we can take the weighted distance value as a reference on how much we can trust the result since when bad results are obtained this weighted distance value is usually very low. Please refer to the online results in order to take a better look at the insights of the word matches.



Figure 7. Task: "find a sport with no ball"

IV. CONCLUSIONS AND FUTURE WORK

Our system processes tasks issued to a robot to search and find objects by its description and look for its matches from different scenes. We use word representations in vector spaces to expand the semantic scope of the descriptions and improve the matching between them. It has been proved that our system is able to properly obtain scene relations to a certain descriptive task using the semantic relations between the descriptions and the robot search task. The system can even provide meaningful results when queried with words that don't even directly appear on the scene descriptions. On the other hand some results might not be accurate enough sometimes due to not very accurate semantic relations and other times due to errors on the scene descriptions. Therefore there is room for improvement on both sides. We could take into account the value of the cosine distance and discard the results when values are too low, as we observed that low values always correspond to very bad matches. Also new deep learning techniques can be applied for the scene description generation in order to improve this part of the system. Dynamically selecting the most important parts of the description on the task can provide an improvement as we can give them a different weighs on the semantic matching algorithm. Other word semantic relation techniques can be tested in order to look for a better semantic matching between the task and the scene description.

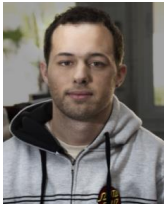
ACKNOWLEDGMENT

This work has been conducted as part of an A*STAR Research Attachment Programme (ARAP) at the Human Language Technology Department of Institute for Infocomm Research, Singapore.

REFERENCES

- [1] X. He, R. Srivastava, J. Gao, and L. Deng, "Joint learning of distributed representations for images and texts," arXiv preprint arXiv: 1504.03083, 2015.
- [2] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," arXiv preprint arXiv: 1411.4555, 2014.
- [3] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," arXiv preprint arXiv: 1502.03044, 2015.
- [4] T. Mikolov, W. T. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *HLT-NAACL*, June 2013, pp. 746-751.
- [5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391-407, 1990.
- [6] M. Sahlgren, "The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces," *Institutionen for Lingvistik*, 2006.
- [7] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," arXiv preprint arXiv: 1309.4168, 2013.
- [8] N. J. Van Eck, L. Waltman, and J. van den Berg, "A novel algorithm for visualizing concept associations," in *Proc. Sixteenth International Workshop on Database and Expert Systems Applications*, August 2005, pp. 405-409.

- [9] G. A. Miller, "WordNet: A lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39-41, 1995.
- [10] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD International Conference on Management of Data*, June 2008, pp. 1247-1250.
- [11] ACMS. Christoph, L. O. F. L., *Measuring Semantic Similarity and Relatedness with Distributional and Knowledge-Based Approaches*, 2016.
- [12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv: 1301.3781, 2013.
- [13] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, O'Reilly Media, Inc. 2009.
- [14] R. Kiros, R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models," arXiv preprint arXiv: 1411.2539, 2014.
- [15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-178, 1997.
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012, pp. 1097-1105.
- [17] S. T. Dumais, "Latent semantic analysis," *Annual Review of Information Science and Technology*, vol. 38, no. 1, pp. 188-230, 2004.
- [18] A. Zhila, W. T. Yih, C. Meek, G. Zweig, and T. Mikolov, "Combining heterogeneous models for measuring relational similarity," in *HLT-NAACL*, 2013, pp. 1000-1009.
- [19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, 2013, pp. 3111-3119.
- [20] B. Plummer, L. Wang, C. Cervantes, J. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," arXiv preprint arXiv:1505.04870, 2015.
- [21] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, et al., "Microsoft COCO: Common objects in context," in *Computer Vision-ECCV*, Springer International Publishing, 2014, pp. 740-755).
- [22] H. Besser, "Visual access to visual images: The UC Berkeley image database project," *Library Trends*, vol. 38, no. 4, pp. 787-798, 1990.



Marco A. Gutiérrez is a PhD student in cognitive vision for robotics systems at the Robotics and Artificial Vision Laboratory (RoboLab) from the University of Extremadura, Spain since 2011. He is currently holding an A*STAR Research Attachment Programme (ARAP) scholarship in the Human Language Technology Department at I2R, A*STAR, Singapore. He obtained the highest evaluation possible

(above expectations on all fields) on his internship at KUKA Laboratories GmbH, Augsburg, Germany in 2012. Recently his team (Ursus) was awarded with Best Team for Functionality Benchmark on Object Perception and Speech Understanding at the Rocking Robot Challenge 2014 in Toulouse, France. He has contributed to several open-source robotics and computer vision related projects such as RoboComp and the Point Cloud Library even as organization administrator and mentor (respectively) for several editions of the Google Summer of Code programme (2013, 2014 and 2015). His recent areas of research include cognitive vision, deep neural networks, multimodal systems and word semantics. He is organizer of the *Workshop on Multimodal Semantics for Robotics Systems* and Advisory Committee for the *The Path to Success: Failures in Real Robots* Workshop that will take place as part of next IROS 2015 conference in Hamburg, Germany.



Rafel E. Banchs (M'14) is currently a Research Scientist at the Institute for Infocomm Research in Singapore. He received his Ph.D. in Electrical Engineering from the University of Texas at Austin in 1998. He was awarded a Ramon y Cajal fellowship from the Spanish Ministry of Education and Science from 2004 to 2009. His recent areas of research include Machine Translation, Information Retrieval, Cross-language Information Retrieval and Dialogue Systems. More specifically, he has been working on the application of vector space models along with linear and non-linear projection techniques to improve the quality of statistical machine translation and cross-language information retrieval systems. He has served as co-organizer of the 2nd TC-STAR Work-shop on Speech to Speech Translation 2006; the First International Workshop on Content Analysis in the Web2.0 (CAW2) at WWW 2009, the ESIRMT-HyTra Joint Workshop at EACL'12, the CREDISLAS workshop at LREC'12, the Special Session "Rediscovering 50 Years of Discoveries" at ACL'12, HyTra-2 and HyTra-3 workshops at ACL'13 and EACL'14, and NII-Shonan Meeting Seminar 059 on "The Future of Human-Robot Spoken Dialogue: from Information Services to Virtual Assistants". He has also served as area chair for IICNLP'11, general co-chair of AIRS'13 and PC chair of IALP'14

Chapter 6

Exploiting Symmetries and Extrusions for Grasping Household Objects

Exploiting Symmetries and Extrusions for Grasping Household Objects

Ana Huamán Quispe¹ Benoît Milville¹ Marco A. Gutiérrez² Can Erdogan¹ Mike Stilman[†]
Henrik Christensen¹ Heni Ben Amor¹

Abstract—In this paper we present an approach for creating complete shape representations from a single depth image for robot grasping. We introduce algorithms for completing partial point clouds based on the analysis of symmetry and extrusion patterns in observed shapes. Identified patterns are used to generate a complete mesh of the object, which is, in turn, used for grasp planning. The approach allows robots to predict the shape of objects and include invisible regions into the grasp planning step. We show that the identification of shape patterns, such as extrusions, can be used for fast generation and optimization of grasps. Finally, we present experiments performed with our humanoid robot executing pick-up tasks based on single depth images and discuss the applications and shortcomings of our approach.

I. INTRODUCTION

The ability to grasp and manipulate objects is an important skill for autonomous robots. Many important tasks, e.g., assisting humans in household environments, require robots to reliably plan and execute grasps on surrounding objects. To generate plans for manipulation tasks, information about the shape of the object is required. A frequent approach to grasp planning is to use a database of polygonal meshes representing the different objects that the robot can manipulate [8]. Such information about object geometry can be used by grasp planners to synthesize an appropriate hand shape and orientation for physical interaction. While this approach is valid for structured domains with a small set of different objects, it does not scale to unstructured environments in which many objects may have never been seen before.

Other approaches to grasp planning employ depth cameras to acquire 3D point clouds of new objects, which in turn are used to generate grasps. Since the point clouds are acquired from a specific perspective, they only hold partial shape information about the visible frontal part. Using only partial point clouds to plan manipulation tasks can be very limiting, since many grasps involve placing fingers on opposite sides of an object. To fill any gaps and produce a complete point cloud, multiple images can be acquired by either iteratively moving the camera or the object. This process is time-consuming and

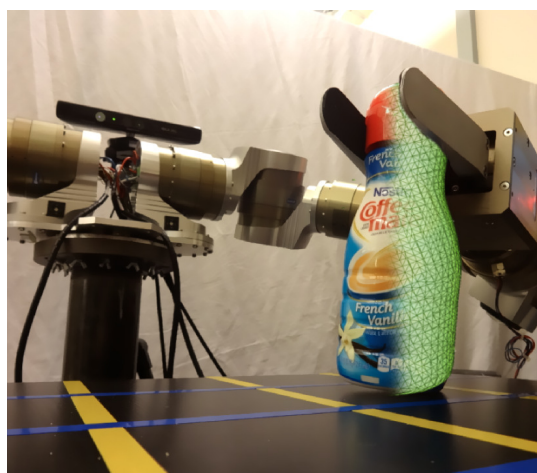


Fig. 1: Extracted information of rotational symmetries in the object is used to create a complete shape from a partial point cloud. The generated mesh is used by a grasp planner to generate a continuous set of grasps around the symmetry axis.

introduces new challenges such as the precise matching of the individual point clouds of each view.

Alternatively, the robot can use geometric cues to predict the shape of the object in unseen regions. Through the analysis of inherent shape properties such as mirror symmetries and rotational extrusions, estimates of the complete point cloud can be generated from a single image. The extracted symmetry parameters can be used to extend observed shape patterns, e.g., the profile curve of an object, to occluded regions.

In this paper, we show how compact object representations for manipulation tasks can be generated from a partial point cloud. Given a *single* RGB-D image, we generate a complete mesh model of the observed object as well as additional shape information, e.g., axis of symmetry or superquadric approximations. We show that these compact representations can be later exploited for the fast synthesis of a continuous set of grasps. In turn, the set is used to plan robot manipulation tasks. Our approach builds both upon recent developments in symmetry-based [3, 18], as well as extrusion-based object representations [16]. Symmetry-based representations mirror observed object parts into occluded regions. Extrusion-based approaches, on the other hand, try

¹Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA. ahuaman3@gatech.edu, cerdogan@cc.gatech.edu, benoit.milville@gadz.org, hic@cc.gatech.edu, hbenamor@cc.gatech.edu

²Robotics and Artificial Laboratory, University of Extremadura, Cáceres, 10003, Spain. marcog@unex.es

to identify a two-dimensional profile which can be linearly or rotationally extruded to complete an object. In this work we show how symmetries and extrusions can be used to extract two different types of object representations, namely superquadric approximations and 2D shape profiles. We also show how these representations can be used to generate grasps on the object.

The rest of this paper is organized as follows: Section II summarizes relevant literature. Section III introduces two compact object representations that are based on detecting symmetries and extrusions. Section IV shows how compact object representations based on extrusion patterns can be exploited for fast grasp planning with a small number of parameters. Section V presents experimental results of the object completion, as well as its application to robot grasping tasks. Finally in Section VI we discuss our approach and its advantages and shortcomings.

II. RELATED WORK

For a robot to physically interact with its environment, algorithms for both grasp planning and perception are required. Traditional approaches for grasp generation are often based on fitting 3D CAD models to the observed scene [14, 15]. Such an approach, however, cannot be used to grasp novel objects since it requires accurate, prior knowledge about the shape. With the advent of depth cameras, various researchers have turned towards point cloud representations for perception and grasp planning. Huebner et al. [11] showed that bounding boxes computed from point clouds can be used to grasp novel objects. In a similar vein, Jiang et al. [12] proposed a so-called grasping-rectangle representation which can be used to infer the best grasp parameters given an RGB-D image of a novel object (given an offline training step). Przybylski et al. [21] showed simulation results in which a medial axis representation of objects can be used to find successful grasps without compromising on the approximation quality. Other than boxes and spheres [17], superquadrics [9] have also been considered for grasping applications given their compactness and ability to represent many diverse shapes with a limited number of parameters. Recently, Duncan et al. presented a fast hierarchical approach to fit superquadrics online [5].

On the side of grasp generation, a popular metric used to predict grasp robustness is the ϵ metric proposed by Ferrari and Canny [6]. While many popular grasp generators, such as GraspIt! use this metric to evaluate and refine the grasp search, it has been noted [4] that a grasp with a good metric does not translate to a robust grasp in a real-world execution. Researchers such as Hsiao [10] and Balasubramanian [1] have shown that grasps obtained using simple human heuristics can produce comparable or even better results when evaluated in a real, non-simulated environment.

A real world scenario - contrary to a simulated one - presents its own set of challenges: errors in perception, control and modeling must be considered and might render an optimal simulated grasp into an infeasible one. Regarding incomplete perceptual information, such as one-view point clouds for a

given object, Bohg et al. [3] proposed a simple approach that exploits the symmetry of most common household objects to predict the full shape of an object on a tabletop scenario.

Following Bohg's observation that most common household objects present similar characteristics (such as symmetry, extrusion-like geometry and primitive shapes), we use them to approximate the shape of objects. This is also useful in the event of occlusion, in which a complete point cloud is not available.

III. GENERATING COMPACT OBJECT REPRESENTATIONS FROM SINGLE RGB-D IMAGES

In this section, we present two compact representations of objects that can be generated from partial point clouds. These representations can be used to plan grasps on objects involving regions of the point cloud that are currently invisible. As a result, a wider range of grasps can be planned, including, for example, side grasps which are based on an opposition of fingers placed at the front (seen) and the back (unseen) of the object.

We will first present a superquadric representation which is based on determining symmetries in point clouds. After that, we will turn towards a more detailed representation which makes use of rotational symmetries and linear extrusions to characterize an object.

A. Superquadric Representation

Superquadrics are a family of geometric shapes that can represent a wide range of diverse objects. The equation describing superquadrics in their canonical form can be written as

$$F(\mathbf{x}) = \left(\left(\frac{x}{a} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{b} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{c} \right)^{\frac{2}{\epsilon_1}} = 1. \quad (1)$$

where a, b, c are the scaling factors along the principal axes, ϵ_1 is the shape factor of the superquadric cross section in a plane orthogonal to XY containing the axis Z, and ϵ_2 is the shape factor of the superquadric cross section in a plane parallel to XY. If a general transformation is considered, then the total number of parameters required to define a superquadric is 11 (the 6 additional being the rotational and translational degrees-of-freedom (DoFs) $\{x, y, z, \rho, \psi, \theta\}$). By minimizing the error between each point and the general superquadric equation, a shape that best fits the point cloud can be obtained:

$$\min_k \sum_{k=0}^n \left(\sqrt{abc} F^{\epsilon_1}(\mathbf{x}; \Lambda) - 1 \right)^2 \quad (2)$$

As mentioned in Section II, superquadrics have previously been used to generate grasp configurations for simple objects [2, 22]. Most of these approaches assume that the complete shape of the object is given or that the parameters can be learned beforehand. However, when working with depth cameras this is not a reasonable assumption to make. In recent work, Duncan et al. [5] presented a superquadric fitting

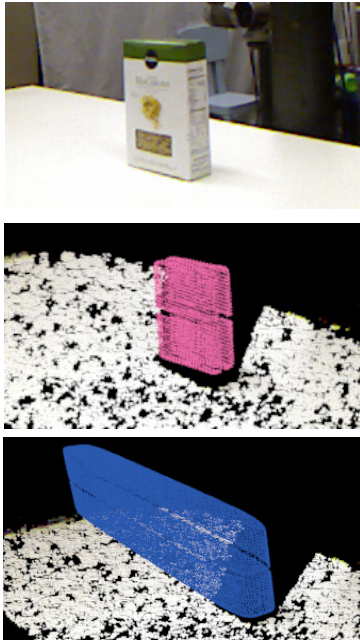


Fig. 2: An example for the superquadric fitting with symmetry analysis (middle) and without it (bottom).

approach which uses a voxel representation to reduce the computational complexity of the task. We found that this approach worked well when the segmented point cloud of the object had a good viewing point (i.e. the front, side and top of the object were seen). For point clouds in which only one side of the object was seen (i.e. only front), the performance quickly deteriorated, producing fitting parameters that in many cases exceeded greatly the original dimensions of the objects. While this could be partially alleviated by hard-coding limits in the dimension of the axes, this is not practical when dealing with novel objects, for which we might not know the dimensions beforehand.

Inspired by work presented by Bohg et al. [3], we added an additional pre-processing step to the superquadric minimization process. Instead of using the original point cloud as input, we generated a mirrored version (see Fig. 2) by finding an optimal symmetry plane perpendicular to the table where the object resides (for more details of this process, please refer to the original paper [3]).

B. Object Completion from Extrusions

Planning task-specific grasps requires information about the complete shape of the object to be manipulated. Many household objects are based on extrusions. Indeed many modelling and manufacturing systems use linear and rotational extrusions in a hierarchy to generate the models used for manufacturing. Uncovering extrusions in partial point clouds can therefore help to generate a complete point cloud from a partial observation. In addition, this knowledge can be used to create a large set of feasible grasps from which a planner can

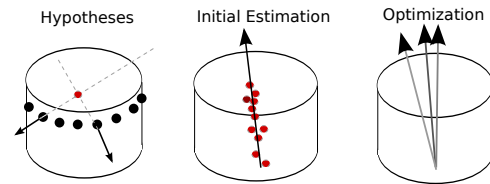


Fig. 3: The three steps used for optimizing the axis of extrusion. First, we generate hypotheses by analysing pairs of points. The resulting estimates are used to produce an initial estimate of the axis of extrusion. Finally, optimization is used to improve the extrusion axis.

choose suitable candidates for task execution. For example, detecting the axis of symmetry in a rotationally symmetric object allows us to rotate any feasible grasp around this axis.

In this paper, extrusion detection is performed using a three-step approach, see Fig. 3 for an overview of the approach using rotational extrusions. In the first step, we use points from the partial point clouds to generate hypotheses for the extrusion axis. In the case of rotational extrusions, we randomly sample pairs of points and use the normal of each point to create a line. Each pair of lines is intersected and the resulting point is used as a hypothesis for the axis of extrusion. Fig. 3 shows an example for points sampled from a cylindrical object. To account for noise, we use the midpoint of the line connecting the closest points, in case the two lines do not intersect. The collected hypotheses points are then used to create an initial estimate of the axis of extrusion. To this end, we fit a line into the set of hypotheses using linear least-squares. The RANSAC [7] algorithm is further used to reduce the influence of outliers. Given this initial estimate, we perform optimization to produce a more accurate axis of extrusion. Specifically, we use the dynamic hill climbing algorithm [23] to search for an axis of extrusion which reduces the dispersion of points along the profile of the object. In every iteration, the axis of extrusion is used to rotate all points of the partial point cloud back onto a plane. We then estimate the density of the points using a kernel density estimator [20]. By maximizing the density using the hill climbing algorithm, we can reduce the dispersion of the projected points, thereby recreating the profile of the object. However, performing a kernel density estimation in each step of the optimization process is computationally expensive and does not scale to large point clouds. The following method is, therefore, a discrete approximation of the kernel density, which produced accurate results in practice while at the same time being fast.

We create an approximation of the kernel density estimator by creating a grid over the projected point cloud. The number of cells used in our experiments varied between 5 and 30 cells in each dimension. For each cell $i \in \{1, \dots, M\}$ we count the number of points c_i that lie within. We then calculate the average of the differences to neighbouring cells $j \in \{1, \dots, N\}$. The overall objective function of the optimization can be

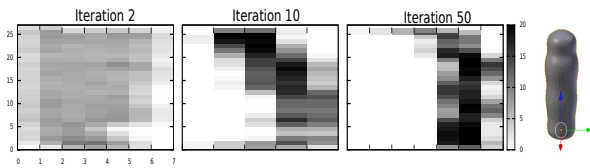


Fig. 4: Density estimation at different stages of optimization. At the beginning of the optimization, the projected points are highly dispersed. The axis of extrusion is then changed to minimize the dispersion, such that the outer profile of the object emerges as can be seen in iteration 50. On the right side we can see the object to which the profile belongs.

written as

$$E = \frac{1}{M} \frac{1}{N} \sum_i^M \sum_j^N \|c_i - c_j\| \quad (3)$$

where E is the energy to be minimized. Fig. 4 shows three iterations during the optimization of the axis of extrusion. Dark areas correspond to regions of high density of points, while lighter areas correspond to low density regions. In early iterations, the estimate of the axis does not produce a clear profile when points are projected (rotationally) onto a plane. In iteration ten, we can see that high density regions start forming. After fifty iterations, an approximate profile of the object starts to emerge.

After optimization is finished, we regard the projected points as the profile of the object and rotationally extrude them around the axis of extrusion to generate a complete point cloud. Fig. 5 shows a set of household objects, the recorded depth images, as well as the reconstructed complete meshes. Given the completed point cloud, we reconstructed the meshes using Poisson surface reconstruction [13].

For the case of linear extrusions along an axis, a different method for the estimation of the initial axis of extrusion needs to be used. For linear extrusions, we compare the normal vectors of pairs of points and generate a hypothesis if the difference between the normals is below a threshold. The resulting set of hypothesis can then be clustered, such that each cluster represents a possible axis of extrusions. For example, for a box, up to six clusters can be found.

Note, that in our approach we use a point cloud to represent the profile of an extrusion. For revolute objects, the profile defines the outer curve of the object, which can be rotated around the axis of extrusion to generate the complete shape. For linear extrusions, the cloud represents the basic 2D shape which can be extruded to form the object. Fig. 6 shows the extracted object profiles for objects with linear extrusions.

IV. USING COMPACT OBJECT REPRESENTATIONS FOR GRASP PLANNING

Grasp planning greatly benefits from the completed point clouds. A complete point cloud can be triangulated and used as an input to existing grasp generation and planning algorithms. In contrast to the partial point cloud, the completed and triangulated mesh can be used to perform collision checks

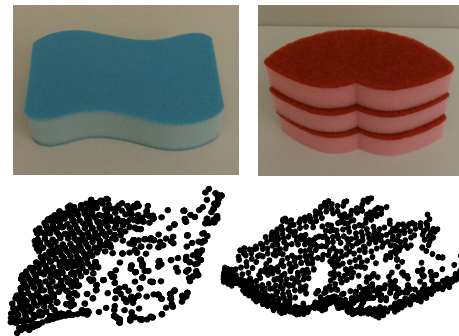


Fig. 6: Extracted object profile for the linearly extruded objects. The extracted profiles are used to create a complete point cloud.

and evaluate grasp quality using existing metrics. In contrast, traditional grasp quality metrics cannot be directly applied to partial point clouds. Similarly, having a complete mesh allows a grasp planner to evaluate a large variety of grasps, which can then be pruned based on task constraints. However, generating many grasps often involves repeated applications of grasp optimization methods which can be computationally demanding, in particular in the presence of many degrees-of-freedom in the robot arm and hand. Extracted shape information from extrusions can be used to improve the efficiency of this process by significantly reducing the number of degrees-of-freedom of the problem.

The main insight of this section is that hand shapes during object grasping are invariant to movements along the axis of extrusion. As long as the robot hand moves along the axis of extrusion, no expensive replanning of the hand shape is necessary. In the case of linear extrusions, the robot hand can move up and down the axis of extrusion without having to change the hand shape. Similarly, in the case of rotational extrusions, the hand can be rotated around the axis of extrusion. This knowledge can be exploited during grasp generation in order to turn each single detected grasp into a continuous set of grasps. Subsequently, we present a specific example how information about extrusions can be used to reduce the dimensionality and complexity of a grasp re-planning task.

Fig. 7a shows a scenario, in which a grasp is executed on a rotationally symmetric object. The grasp has a low manipulability index which is not sufficient to achieve the task constraints. Typically, this means that a new grasp and arm pose needs to be planned, which involves (sampling-based) optimization in the high-dimensional space of joint angles.

Given that the grasp is performed on a rotationally symmetric object, the grasp generation can be modeled as an inverse kinematics problem where the goal is to determine an arm configuration q that is collision free. The output is constrained by the end-effector position on the object and the corresponding inverse kinematics solution. The end-effector pose x can be parametrized by (1) the rotation around the axis of extrusion ϕ and (2) the distance along the axis of extrusion



Fig. 5: Reconstruction of rotationally symmetric household objects. The top row shows a photo of the object. The middle row shows the corresponding depth image recorded using a Microsoft Kinect. The bottom row shows the completed mesh. Reconstruction was performed from a single image through the analysis of extrusions.

α , $\mathbf{x} = \text{pose}(\phi, \alpha)$. The inverse kinematics solution \mathbf{q} with a 7-DoF arm for an end-effector pose \mathbf{x} can be parametrized by an additional variable θ which represents the angle between the wrist-elbow-shoulder plane and the ground, $\mathbf{q} = IK(\mathbf{x}, \theta)$.

At each iteration i , the new arm position is computed using an updated grasp position from the parameter space $\{\phi_{i-1} \pm \delta_\phi, \alpha_{i-1} \pm \delta_\alpha\}$ and the corresponding inverse kinematics parametrized by $\{\theta_{i-1} - \delta_\theta, \theta_{i-1}, \theta_{i-1} + \delta_\theta\}$. Let \mathbb{P} represent the full space of the variables ϕ , α , and θ . The algorithm iteratively updates these parameters by determining which tuple leads to the maximum manipulability [19]. This is realized by solving for the following objective

$$\mathbf{q}_i = \underset{\{\phi, \alpha, \theta\} \in \mathbb{P}}{\text{argmax}} \sqrt{\det(J(\mathbf{q})J^T(\mathbf{q}))} \quad (4)$$

where $\mathbf{q} = IK(\text{pose}(\phi, \alpha), \theta)$. The sequence in Fig. 7 shows several snapshots during this optimization. In this scenario, the robot grasps a rotationally symmetric bottle. The initial random grasp sample in Fig. 7a yields a manipulability of 0.268 which is then improved in Fig. 7d leading to a value of 0.540. To optimize the manipulability, the planner iteratively changes the grasp position on the robot with the ϕ and α parameters, and the inverse kinematics parameter θ . This optimization can be performed efficiently since, the high-dimensional configuration space of the hand does not need to be represented thanks to the extracted symmetries. Instead, a three-dimensional space of parameters $\{\phi, \alpha, \theta\} \in \mathbb{P}$ is used.

V. EXPERIMENTAL RESULTS

In this section, we present a set of experiments which we conducted to evaluate the proposed approach. The first

set of experiments focuses on the complexity and accuracy of point cloud completion when generating compact object representations. The second set of experiments shows the application of the approach to grasp planning on a humanoid robot. The used humanoid robot is based on Schunk LWA3 arms with 7 DoF. A Schunk gripper with a maximum aperture of 7cm was used. Partial point clouds were recorded using a Microsoft Kinect camera.

A. Accuracy of Fit

We first analyzed the accuracy of fit of the two presented compact object representations. For extrusions, we collected a set of rotationally symmetric meshes from internet databases from which we generated partial point clouds. We then cut out a partial point cloud representing 30% of the data and simulated Kinect-like noise by adding holes and noise to the dataset. The partial cloud was then completed using the extrusion detection methods from Sec. III-B. To measure the accuracy, we compared the completed clouds to the original mesh of the object. On average, the approach produced an error (distance of points to mesh) of 2mm, where objects had a diameter between 10 – 20cm. Analysis of the extrusions required on average 200ms.

For superquadric fitting we conducted a similar experiment. However, in this case we noticed larger variations in the reconstructed shapes depending on the perspective of the camera to the object. We therefore placed each object at one of five different locations in front of the camera and measured the run time of the algorithm including symmetry analysis and without it. As depicted in Tab. I, the fitting time is shorter when additional points are added via symmetry analysis. While this

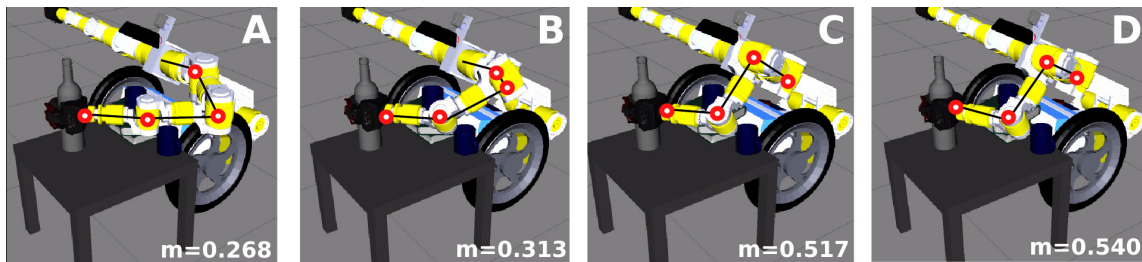


Fig. 7: Grasp manipulability optimization along the axis of extrusion. Since the object is symmetric, the same hand configuration can be rotated around the object (A-B, C-D). At the same time, the extra DOF in the inverse kinematics solution is also utilized to maximize manipulability (B-C).

may seem unintuitive, we found that the superquadric shape has more constraints when considering mirrored points. As a result, the optimization process required for fitting quickly settles on a good solution.

TABLE I: Comparison of fitting times

Object	Input	P1	P2	P3	P4	P5	Avg. Time
Apple	Symmetry	0.02	0.13	0.01	0.06	0.07	0.05s
	Plain	0.14	0.17	0.06	0.06	0.06	0.098s
Milk	Symmetry	0.20	0.07	0.03	0.05	0.04	0.078s
	Plain	0.42	0.56	0.27	0.53	0.06	0.368s
Jam	Symmetry	0.06	0.11	0.13	0.08	0.21	0.118s
	Plain	0.08	0.29	0.10	0.08	0.11	0.132s
Raisins	Symmetry	0.29	0.25	0.31	0.14	0.27	0.252s
	Plain	0.36	0.40	0.43	0.43	0.32	0.388s
Creamer	Symmetry	0.15	0.15	0.22	0.13	0.14	0.158s
	Plain	0.65	0.09	0.39	0.26	0.29	0.336s

B. Robot Grasping Experiments

Next, we conducted an experiment in which a humanoid robot was used to grasp household objects located in front of it. We also placed several other objects as clutter on the table. Given the depth image all objects were reconstructed using compact object representations. After that, the robot planned and executed grasps using the normal at a point as an approach direction and the method described in Sec. IV for ensuring manipulability and obstacle avoidance. We conducted trials with 4 objects which were placed at 4 different locations on the table. Each trial was repeated three times. A grasp was regarded successful if the robot was able to lift the object.

Tab. II summarizes the results of the experiment. We can see that the approach using superquadrics performs well on most objects with the exception of the roll. In contrast, the extrusion-based approach seems to have difficulties with a specific location (C3). Analyzing the robot executions, we found that superquadric approach typically leads to approximate shapes which are slightly larger than the original object. Hence, the executed grasp includes a "buffer" zone that allows it to succeed in the presence of sensor and calibration noise. Grasps planned for the shapes generated by the symmetry detection, however, are tightly fit to the object. This often lead to premature contact with the object during grasp execution. In Tab. II we also see the number of different grasps found using the two approaches. We can see that the symmetry based approach leads to a larger number of different grasps, due

to the invariance along the axis of extrusion. Images of the executed grasp and the experimental setup can be found in Fig. 8.

TABLE II: Experimental results, 3 trials per object per location

		Location	Creamer	Dove	Roll	Micro
Success	Extrusion	B4	100%	100%	0%	100%
		C3	0%	0%	0%	0%
		C4	100%	100%	100%	100%
	SQ	B4	100%	100%	0%	100%
		C3	100%	100%	66%	100%
		C4	100%	100%	0%	100%
Grasps	Extrusion	B4	1040	900	1200	640
		C3	800	400	2200	800
		C4	1270	320	800	1020
	SQ	B4	11	11	7	11
		C3	7	3	1	7
		C4	10	5	5	13

VI. DISCUSSION AND CONCLUSION

In this paper we introduced methods for generating compact and complete object representations that are particularly useful for robot grasping applications. The approach exploits natural patterns found in many shapes, e.g., symmetries, linear extrusions, and rotational extrusions to generate a complete mesh from a single depth image. We also showed that the extraction of this information can be used to improve the efficiency and quality of the grasp planning step. The work presented in this paper can be seen as a first step towards shape priors that can be used by a robot to generate hypotheses about the shape of an object in invisible regions. Other cues, such as curvature and texture may also be helpful in predicting the complete shape from partial observations. At the moment the introduced approach is limited to household objects, which are often based on linear and rotational extrusions. However, it can also be extended to work in a hierarchy to complete more complex objects. In future work, we hope to investigate this aspect in more detail.

The performed robot experiments showed that the approach can be used to create a variety of grasps. In particular, we can generate grasps that extend to parts of the object that are not seen. This is in contrast to other methods which limit the approach direction of the robot to the visible part of the object. We have shown in the experiments that the method can be used to reconstruct objects in a cluttered scene without prior

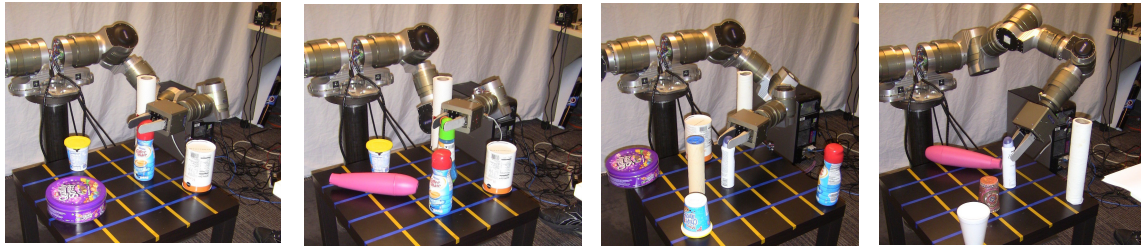


Fig. 8: Grasps on household objects generated via grasp planning on compact object representations. All objects on the table were reconstructed. Objects that were not grasps were regarded as obstacles to be avoided during the manipulation task.

information. Yet, the additional information gained by creating complete meshes also imposes additional requirements on the accuracy of the robot controller. Planning grasps with more accurate reconstructions of the observed object means that the robot needs to be very precise in the task execution. So far, we do not have a model of the inherent sensor and actuation noise. We hope to investigate Bayesian approaches to object fitting, which would allow us to use information about the uncertainty during task execution.

ACKNOWLEDGMENTS

This work is dedicated to the memory of Mike Stilman, whose enthusiasm for making robots do cool things will always be remembered.

REFERENCES

- [1] R. Balasubramanian, L. Xu, P. D. Brook, J.R. Smith, and Y. Matsuoka. Human-guided grasp measures improve grasp robustness on physical robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010.
- [2] G. Biegelbauer and M. Vincze. Efficient "3d" object detection by fitting superquadrics to range image data for robot's object manipulation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2007.
- [3] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratal, N. Bergstrom, D. Kragic, and A. Morales. Mind the gap: Robotic grasping under incomplete observation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.
- [4] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2010.
- [5] K. Duncan, S. Sarkar, R. Alqasemi, and R. Dubey. Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.
- [6] C. Ferrari and J. Canny. Planning optimal grasps. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1992.
- [7] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.
- [8] C. Goldfeder and P. Allen. Data-driven grasping. *Autonomous Robots*, 2011.
- [9] C. Goldfeder, P. Allen, C. Lackner, and R. Pelossof. Grasp planning via decomposition trees. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2007.
- [10] K. Hsiao, S. Chitta, M. Ciocarlie, and E. Jones. Contact-reactive grasping of objects with partial shape information. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.
- [11] K. Huebner and D. Kragic. Selection of robot pre-grasps using box-based shape approximation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.
- [12] Y. Jiang, S. Moseson, and A. Saxena. Efficient grasping from rgb images: Learning using a new rectangle representation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.
- [13] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proc. of the Fourth Eurographics Symposium on Geometry Processing*, 2006.
- [14] U. Klank, D. Pangercic, R.B. Rusu, and M. Beetz. Real-time cad model matching for mobile manipulation and grasping. In *9th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2009.
- [15] D. Kragic, A. Miller, and P. Allen. Real-time tracking meets online grasp planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2001.
- [16] O. Kroemer, H. Ben Amor, M. Ewerton, and J. Peters. Point cloud completion using extrusions. In *Int. Conf. on Humanoid Robots (Humanoids)*, 2012.
- [17] A. Miller, S. Knoop, H. I. Christensen, and P. Allen. Automatic grasp planning using shape primitives. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2003.
- [18] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM SIGGRAPH 2006 Papers, SIGGRAPH '06*, pages 560–568, New York, NY, USA, 2006. ACM.
- [19] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement and control*, 1986.
- [20] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 1962.
- [21] M. Przybylski, T. Asfour, and R. Dillmann. Unions of balls for shape approximation in robot grasping. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.
- [22] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.
- [23] D. Yuret and M. de la Maza. Dynamic hill climbing: Overcoming the limitations of optimization techniques. In *Second Turkish Symposium on Artificial Intelligence and Neural Networks*, 1993.

Chapter 7

SPAM for a Manipulator by BNM in Unknown Environments

Simultaneous Planning and Mapping (SPAM) for a Manipulator by Best Next Move in Unknown Environments

Dugan Um¹, Marco A. Gutiérrez², Pablo Bustos² and Sungchul Kang³

Abstract—In this paper, we propose a SPAM (Simultaneous Planning and Mapping) technique for a manipulator type robot working in an uncertain environment via a Best Next Move algorithm. Demands for a smart decision to move a manipulator such as humanoid arms in uncertain or crowded environments call for a simultaneous planning and mapping technique. We assume no a priori knowledge of either the obstacles or the rest of the environment exists. For rapid map building and path planning, we use a skin type setup based on 3D depth camera sensors that completely encompass the entire body of a manipulator. The 3D sensors capture the point clouds used to create an instantaneous c-space map whereby a Best Next Move algorithm directs the motion of the manipulator. The Best Next Move algorithm utilizes the gradient of the density distribution of the k-nearest-neighborhood sets in c-space. It has tendency to travel along the direction by which the point clouds spread in space, thus rendering faster mapping of c-space obstacles.

The proposed algorithm is compared with several sensor based algorithms for performance measurement such as map completion rate, distribution of samples, total nodes, etc. Some improved performances are reported for the proposed algorithm. Several possible applications include semi-autonomous tele-robotics planning, humanoid arm path planning, among others.

I. INTRODUCTION

Motion planning in unknown environments is a challenging problem in path planning. Sensor based approaches have been the dominant trends in the study of unknown environment planning for decades. When it comes to unknown environment planning, a planner calls for continuous perception and planning, thereby closing the loop between sensation and actuation. Due to the limited sensing distance of most of the depth sensor or visual occlusion, only the local area is known to the robot for local path planning. No optimum global path generation idea is reported so far due to the uncertainty innate by an unknown environment. However, if a planner can produce a global map rapidly, optimum path planning is feasible in unknown environment.

Sequential mapping of a local area and path planning is a natural step for sensor based motion planning. In [1], a novel framework for an unknown environment path planning of a manipulator type robot is proposed. The framework described in [1] is a sensor based planner composed of a sequence of multiple MBPs (Model Based Planners) in the

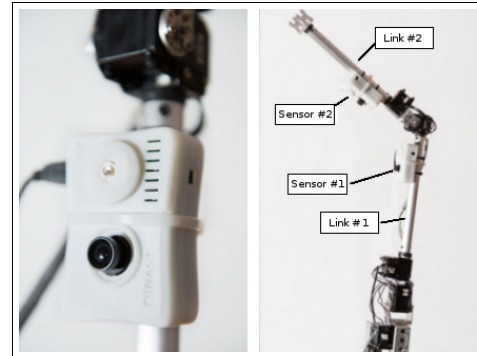


Fig. 1. IPA sensor installation on a three Degrees Of Freedom (3-DOF) robotic linkage

notion of cognitive planning using realtime rehearsal. C-space Entropy is examined in [2] for planning and exploration for a robot with an eye-in-hand sensor system. Natural planning and expanding steps of the c-space is repeated in the paper for an unknown environment path planning algorithm. However, the eye-in-hand sensor has limitation in reporting collision or c-space mapping in realtime due to visual occlusion.

Other studies in sensor based planning of manipulators include [3] whereby an avoidance ability of a redundant manipulator is studied with a moving camera on the ceiling. In [4], trajectory planning for a redundant mobile manipulator is studied using avoidance manipulability concept. Manipulability is the point of study for best path selection in multiple available configurations with singularity and manipulability ellipsoid. In [5] and [6], kinematic analysis in local or global motion planning for a manipulator has been studied in the notion of singularity for a redundant robot as well. Topological analysis in conjunction with singularity concern for a redundant manipulator is dealt with the study on critical point surfaces in configuration space. In summary, the result in the paper implies that a manipulator has to stay in a continuous c-sheet to avoid singularity, which means that a motion planning with inverse kinematic concern is neither robust nor efficient due to the limited utility of a given c-space.

As a result, majority of the manipulator planning schemes we investigated are either hindered by the sensor configuration or by motion constraints for a realtime manipulator motion planning especially for a crowded unknown environments. More flexible sensor configurations for maximum

¹D. Um is with Texas A&M University Corpus Christi, USA
dugan.um at tamucc.edu

² M. A. Gutiérrez and P. Bustos are with Robolab, University of Extremadura, Spain marcocg at unex.es

³S. Kang is with Korea Institute of Science and Technology (KIST)
kasch at kist.re.kr

coverage in conjunction with a supportive planning algorithm is, therefore, in essential need.

To that end, we propose a skin type sensor made out of cameras with 3D depth sensing capability to tackle an unknown environment manipulator motion planning problem (see 1 for a 3-DOF robot example). Our approach is a probabilistic path planning with Simultaneous Planning and Mapping, thus SPAM. For rapid map building and path planning, we use the skin type sensors that completely encompass the entire body of a manipulator. Such sensor can generate realtime point clouds of obstacles from any posture of the robot; thereby a realtime local c-space construction becomes feasible. However, an appropriate guidance algorithm of the robot in global motion planning is of utmost importance for maximum exploration and convergence capability. To that end, we envision a 3D point cloud registration method a possible guidance algorithm for manipulators motion.

3D point cloud registration calls for various descriptors for object recognition [7]. We take advantage of such registration processes to propose a guidance method of a manipulator in a partially constructed c-space environment. Amongst many registration methods, is the Group Averaging Features, an invariant feature for point cloud registration [8], in which a gradient of the density distribution is used to log essential points for 3D shape identification.

We discuss rationale of why and how we utilize the steps of the invariant feature extraction method for sensor based motion planning in Section II. We discuss results of the comparison between simulations of the proposed algorithm and another sensor based planning algorithm in Section III. Map completion rate, distribution of samples, and total nodes are measured for comparison. Section IV shows a real device used to validate the proposed method. Finally in section V improved performances are reported for the proposed algorithm along with some lines of interesting future work.

II. BEST NEXT MOVE PLANNER

Due to the higher order complexity of the manipulator type robot path planning, probabilistic sampling based search is common in general. Several sampling based path planners are reported to be a complete planner so that they either find a path or terminate otherwise. Manipulator path planning in unknown environment, however, is challenging in that neither the path optimality nor the planners completeness can be guaranteed.

Gradual but rapid construction of the c-space map, if feasible, allows a planner to complete a mission in path search with higher probability. C-space mapping especially in a crowded environment is the most daunting task in manipulator path planning though. In [9], Best Next View (BNV) in conjunction with a sensor-based roadmap planner is used for object mapping in unknown environment. Utilization of BNV in the object recognition in an unknown environment is through the concept of detecting key events in the set of range data such as discontinuity of the range data in the scene. These key events are used to drive the global motion

of the manipulator to reduce the ignorance level of the given workspace.

Similarly, we propose the Best Next Move algorithm as a guidance strategy of the robots global motion in uncertain environment. By the BNM algorithm, the local motion in each step is designed to reveal the maximum environmental map possible. We use the point cloud registration scheme in [8] as the best next move strategy since it calls for rapid point cloud identification and collection of a 3D shape.

When it comes to unknown environment manipulator motion planning, two subjects have to be addressed in parallel: map construction and navigation for convergence. We consider map construction and goal convergence as two separate tasks for unknown environment manipulator planning. The more complete c-space map a planner generates, the better chance of convergence to the goal achieved. To that end, objectives of motion strategy are set such that:

- 1) Rapid map construction stage: steering global motion to build a maximum environment map
- 2) Goal convergence stage: achieving search completeness

For the first objective we propose a combinatory motion planning of sampling based search and the point cloud registration inspired approach to determine the Best Next Move (BNM) of the global motion. Best Next Move is the direction possibly to collect maximum information of c-space obstacles.

Group Average Feature (GAF) method, one of the 3D point cloud registration methods is designed to search point cloud sets to register the uniqueness of a 3D object as on [8]. With the sensitive skin type sensor, we can construct a workspace of the robot in realtime and map out the c-space obstacle instantaneously. That workspace will then correspond to the point cloud data obtained by the collision shield developed around the robotic manipulator at a certain point of time. First we collect point cloud data by 3D depth sensors attached on the manipulator.

In order to maximize the benefit of the GAF point cloud registration scheme, we propose a directional navigation of the point automaton in c-space similar to the kernel function to extract features. To that end, for a given P_c (workspace point cloud) at an instance, we propose a virtual c-space sensor model with which, the point automaton in c-space senses c-space obstacles. The virtual sensor in c-space is assumed to have a sensing range, r , and FOV (Field of View), θ , thus it forms a hyper conical sensing range in n-DOF c-space (see Figure 2).

The sensitive skin-setup of sensors covers the entire body of a manipulator, meaning that we can obtain 3D point data all around the robot manipulator. Thanks to that, a collision shield is formed and point clouds of all the obstacles in the workspace at a time t will be collected such that:

$$P_s^t := \{p_j^i \in R | j = 1, \dots, N\} \quad (1)$$

where, p_j^i is the point cloud from sensor j situated at

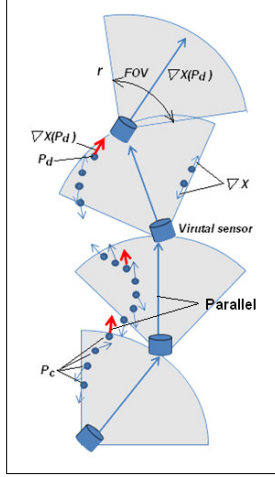


Fig. 2. Algorithm 1&2 at a glance

link i , R is the workspace, N is the total number of point clouds collected by all sensors. Therefore P_s constitutes the set of all point clouds obtained by the robotic manipulator installed sensors at time t . For a given set of joint space variables $\theta_1, \theta_2, \dots, \theta_n$, that define a certain configuration of the manipulator, the forward kinematic model provides transformation matrix such that:

$$T_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} \quad (2)$$

where, R_i is the rotational matrix and t^i is the translational vector for each i_{th} link respectively. Then by the forward kinematics, P_w^t , the point cloud of the collision shield at a certain time t in the workspace coordinate, becomes:

$$P_w^t = \bigcup_j^N P_j^i * T_i \quad (3)$$

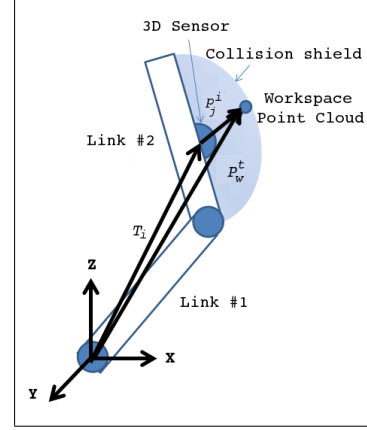
Finally, P_w^t becomes the set of points formed by all the points from the sensors point clouds at time t translated and rotated by its corresponding i_{th} link T_i matrix to the workspace generic coordinates. An example of this in a two link manipulator is shown at Figure 3

For a given point cloud set, P_w^t , at an instance, we create a local c-space map in realtime using RRT (Rapid expanding Random Tree), one of probabilistic sampling algorithm, such that $P_w^t \xrightarrow{RRT} P_c$, so that:

$$P_c := \left\{ p_n \in C^k | n = 1, \dots, M \right\} \quad (4)$$

where, C is the c-space for the robot, k the degree of freedom of the c-space, and M is the number of point clouds produced by the mapping process.

Now we define an 'intensity function' $X : C^k \rightarrow C$ indicating the presence of the point in c-space. We choose to


 Fig. 3. Two link manipulator showing the relationship between the set of point clouds P_s^t and the final resulting one, P_w^t

represent the point set P_c as the sum of overlapping Gaussian distributions. The function X at point $p \in P_c$ is defined as:

$$X(p) = \sum_i \exp \left(- \left(\frac{\|p_i - p\|}{\sigma_G} \right)^2 \right) \quad (5)$$

The gradient of the X is then:

$$\nabla X(p) = \frac{-2}{\sigma_G^2} \sum_i (p_i - p) \exp \left(- \left(\frac{\|p_i - p\|}{\sigma_G} \right)^2 \right) \quad (6)$$

To get an intuition for the meaning of the density gradient, please refer to [8]. For point cloud registration, they further develop kernel functions for object identification. We use the most dominant gradient vector as a guidance to direct the global motion of the point automaton in c-space. Then the planner will steer the point automaton along the most dominant gradient to maximize exploration capability, thus rapidly searches c-space obstacles. The planner may look similar to potential field planner because it steers the robot along the gradient of the cloud density. However, it is different in that it does not always move away from the obstacle, but it has tendency to travel along the direction by which the point clouds spread in space, thus faster mapping of c-space obstacles possible.

The point cloud data, then, becomes following:

$$p_i = [x_i, y_i, z_i, \nabla X_i]^T \quad (7)$$

Total framework of the proposed SPAM cycle is shown in Figure 4. Note that no inverse kinematic solution is necessary for the planner, thus the algorithm is simple and robust. Detail algorithm of c-space mapping is shown in Algorithm 1.

σ_d in Algorithm 1 is the Standard Distance Distribution as shown in Equation 8. Λ_n , occupied c-space or a collection of c-space point clouds at step n will be added to the c-space point clouds, P_c^n , at the end of each mapping loop.

Algorithm 1: c-space mapping

```

// Initialize RRT tree for expansion:
 $\Lambda_n \leftarrow \emptyset;$ 
 $T_{RRT}^n \leftarrow \emptyset;$ 
// workspace point clouds:
 $P_w^n \leftarrow P_w^{n-1} \cup \sum_{i=1}^n (P_i^s \cdot R^i + t^i);$ 
do while  $\sigma_d(\Lambda_n) > \delta_\Lambda$  do
  grow  $T_{RRT}^n$  forward  $\nabla X_n(P_d)$ ;
   $p_c \leftarrow$  a branch grown from  $T_{RRT}^n$ ;
  // if robot collides with workspace
  point cloud:
  if  $B_{ROBOT}(p_c) \cap P_w^n \neq \emptyset$  then
    // Collect c-space point cloud:
     $\Lambda_n \leftarrow \Lambda_n \cup p_c$ ;
    remove  $p_c$  from  $T_{RRT}^n$ ;
  end
end
// Update c-space point cloud:
 $P_c^n = P_c^{n-1} \cup \Lambda_n$ ;
return  $(P_c^n, T_{RRT}^n)$ ;

```

$$\sigma_d = \sqrt{\frac{1}{N} \sum_{i=1}^N (\theta_i - \theta_{mean})^2} \quad (8)$$

Collision check takes place in the virtual workspace by comparing $B_{ROBOT}(P_c)$, the workspace occupied by the robot model, and P_w^n , the point clouds of workspace obstacles. Detail algorithm of Path planning is shown in Algorithm 2.

The condition of the while loop provides the search completeness of the algorithm via space filling. Search continues until either the goal is found, or SDD of the free c-space is too dense ($< \sigma_T$), thus no more exploration is meaningful.

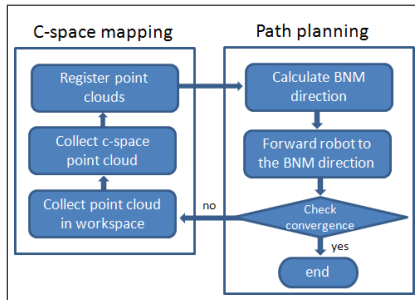


Fig. 4. SPAM cycle

The SPAM cycle is composed of two parallel processes with 5 main functions (See Figure 4). In c-space mapping, main area of work is to convert the workspace point clouds into c-space point clouds by RRT (Rapid growing Random Tree). In a virtual environment, a robot will move by RRT algorithm and whenever collision occurs with a point cloud, the robot's configuration will be sampled and stored as a c-space point cloud data. Based on the registered c-space point clouds in the mapping process, the BNM path planner will guide the robot to the direction obtained by the gradient of the intensity function. Intuitive operation of the algorithm is

Algorithm 2: BNM path planning

```

 $q_i \leftarrow$  initial location for nth expansion;
 $T_{RRT} \leftarrow \emptyset;$ 
 $P_c \leftarrow \emptyset;$ 
do while  $\sigma_d(T_{RRT}) > \sigma_T$  do
  if  $\min d(q, P_c) < \epsilon$  then
    // a path has been found!
    return success
  else
    // N: number of point cloud in  $\Lambda_n$ 
    for  $i \leftarrow 1$  to  $N$  do
       $\nabla X(p) \leftarrow \frac{-2}{\sigma_G^2} \sum_i (p_i - p) \exp\left(-\left(\frac{\|p_i - p\|}{\sigma_G}\right)^2\right);$ 
    end
     $\nabla X_n(p_d) \leftarrow \nabla X_n(p) | \min_{p \in \Lambda_n} d(q_i, \Lambda_n);$ 
  end
  if  $\|q_{i+1} - q_i\| < \epsilon$  then
    // jump to a new free conf with min
    density distribution
     $q_{i+1} = q | \min_{q \in C_f} X(q);$ 
  else
    // initial location for next expansion
     $q_{i+1} = q | \max_{q \in \Lambda_n} d_1(q, q_i);$ 
  end
  call Algorithm1();
  // Update RRT Tree
   $T_{RRT} \leftarrow T_{RRT} \cup T_{RRT}^n$ ;
   $P_c \leftarrow P_c \cup P_c^n$ ;
end

```

shown in Figure 3. As shown in Figure 3, the BNM algorithm has a tendency of directing the point automaton to glide along the surface of c-space objects. This tendency allows rapid search of an object surface so that a systematic unknown environment explorations is feasible.

III. SIMULATION RESULTS

In order to test the proposed algorithm, we setup a 2 DOF revolutionary link robot as a testbed for simulation. Two algorithms are tested for comparison: Sensor-based RRT (see [1] for more detail) and BNM algorithm, introduced in this paper. Using the algorithms the system will try to find a path in an unknown environment with different obstacles in order to reach a certain target. The same sensor model and workspace configurations are applied on both algorithms.

Figures 5(a) and 5(b) show the result of the path search for RTT and BNM respectively. The steps that the robot makes in the different algorithms is shown figures 6(a) and 6(b). For 30 runs of each simulation some statistics are shared in table 1. Total search time is the time in seconds from the beginning to the end of the search operation. No. of P_c stands for the total number of c-space point clouds generated during the search period. The Mapping efficiency in Table 1 is a measure of how efficiently the algorithm generates the c-space map of a given environment. This measure is useful and important since rapid and complete map generation is the key strategy in SPAM in unknown environments. The more information we obtain about the unknown environment, the better the planner can plan a path converging to the goal

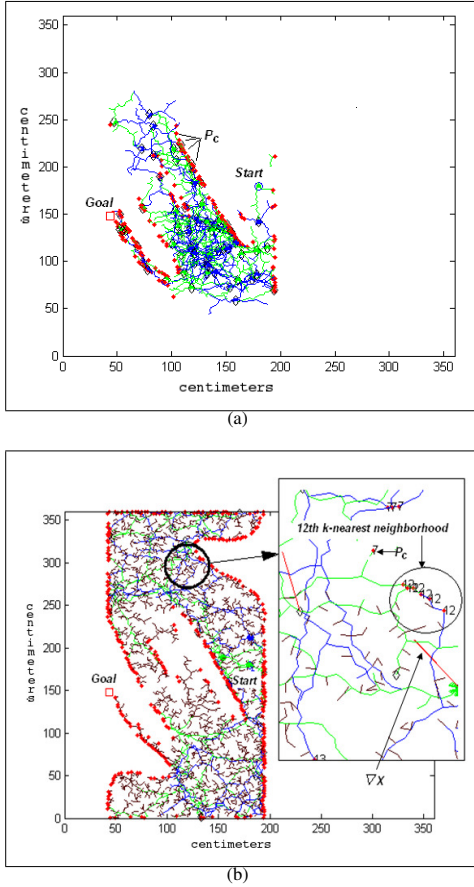


Fig. 5. a) Sensor-based RRT algorithm; b) Sensor-Based BNM algorithm. Each red dot represent a point cloud obtained by the 3D sensors.

point. We define the mapping efficiency such that:

$$\text{Mapping efficiency} = \frac{\% \text{ of map built}}{\text{No. of point clouds in cspace}} \quad (9)$$

With the sensor-based RRT algorithm, about 45% of the c-space map is constructed upon termination, see figure 5(a) for a visual overview and table I for quantified data. Figure 6(a) shows the different positions the robot takes following the RRT algorithm. As milestone in workspace reveals, in the latter figure, overlapping occurs densely in certain areas.

To the contrary, in the second test, BNM algorithm demonstrated about 82% of the c-space construction before the termination (see figure 5(b) and table I). In the magnified window on the right, one can see the red dots that depict 12th k-nearest neighborhood by which the surface of the c-space obstacle is identified. If the window is carefully examined, there are two vectors that show the directions of the gradient of density function. Figure 6(a) shows the milestones in workspace for the robot. You can appreciate that they are more evenly distributed over the entire workspace as a result.

As a summary, overall search time of the sensor-based

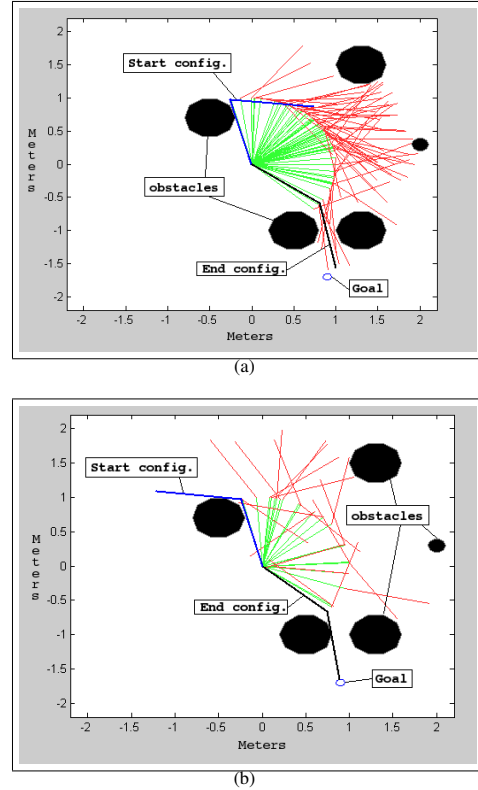


Fig. 6. Milestones in workspace for robot movement by Sensor based: a) RRT algorithm; b) BNM algorithm. Position in blue is the starting position while the black one is the final position, in which the robot reaches the target. Black dots are the obstacles randomly placed in the unknown environment.

TABLE I
SIMULATION RESULTS

Algorithm	Total time	No. of P_c	Mapping efficiency
S-RRT	6205 sec.	1201	45%
BNM	3134 sec.	1144	82%

RRT planner is about twice as much as that of the BNM algorithm. Another thing noticeable, upon the completion of the path search, is the rate of environmental map completion. If you look at figure 5(a) and figure 5(b) for comparison, significantly more environmental map is revealed by the BNM algorithm compared to that of Realtime-RRT planner.

IV. EXPERIMENTS

Two IPA sensors (see [10]) are installed on a manipulator type robot to generate a collision shield around it following the skin type sensor setup (Figure 1). Transformation matrices for local coordinates to the global coordinate for each camera configuration have been setup using Equation 1. The robot tries to reach an object, set as goal, through the unknown environment where an obstacle (a big white box) has been placed. In Figure 7, the robot stops and takes depth map from each sensor, by which a workspace point cloud

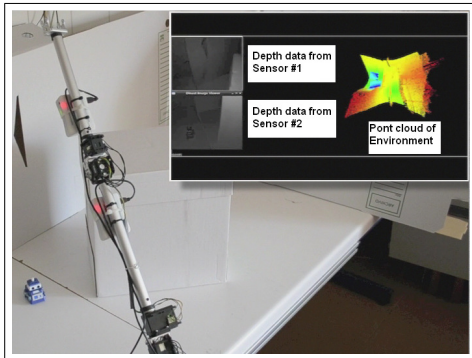


Fig. 7. Point cloud registration with depth data from sensor #1 and sensor #2

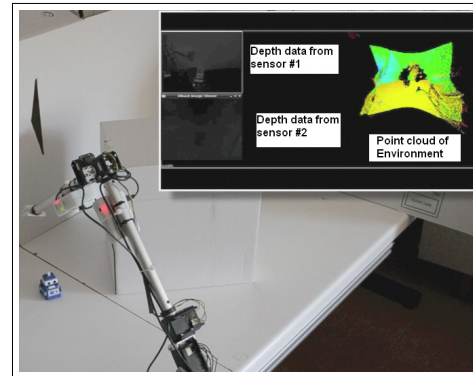


Fig. 9. Robot reaches the goal via BNM algorithm

is generated. With the point cloud map, BNM kicks in to generate a c-space point cloud map as shown in Figure 8. Now the robot is guided along the intensity gradient for the next step generating the maximum environment map until it reaches the goal point (Figure 9).

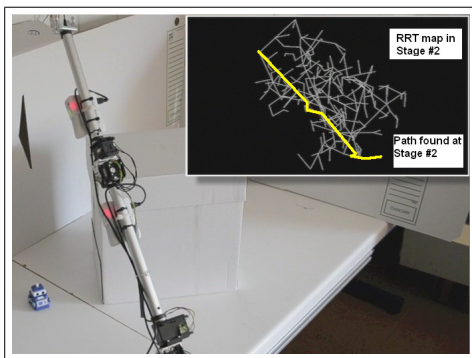


Fig. 8. C-Space point cloud registration by RRT expansion + path planning

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a SPAM (Simultaneous Planning and Mapping) technique for a manipulator type robot working in an uncertain environment via a Best Next Move algorithm. Motivation is in that better map construction capability assures higher success ratio for the convergence to the goal. BNM algorithm offers a means for SPAM of the manipulator planning in uncertain environments thus improving mapping and planning at the same time. For rapid map building and path planning, we use a 3D depth camera based skin type sensors setup that completely encompass the entire body of a manipulator. Captured cloud points by 3D sensors create an instantaneous c-space map whereby a Best Next Move algorithm guides the global motion of the manipulator. We proposed mapping efficiency as a measure of SPAM capability. The proposed BNM algorithm demonstrated up to 82% mapping efficiency in average of 30 runs. As shared in Table 1, BNM not only creates a c-space map with higher mapping efficiency, but also it directs the point

automaton to the goal twice as faster as the sensor based RRT algorithm. We also implemented the BNM algorithm with a sensitive skin sensor setup equipped two linkage manipulator for verification in a real world. Realtime workspace point cloud generation capability from 3D depth sensor data is demonstrated for SPAM technique as well.

The need for the FOV of the camera to cover the entire range of a link could be avoided. Further development of the algorithm in order to make it able to work with less and more spread sensors might end up in great improvements.

REFERENCES

- [1] Dugan Um, Dongseok Ryu, "A Framework for Unknown Environment Manipulator Motion Planning via Model Based Realtime Rehearsal," *Journal of automation, Mobile Robotics & Intelligent Systems*, vol. 5, no. 1, 2011.
- [2] Young Yu, Kamal Gupta, "C-space Entropy: A Measure for View Planning and Exploration for General Robot-Sensor Systems in Unknown Environments," *Int. Journal of Robotics Research*, Vol. 23, No. 12, pp. 1197-1223, Dec. 2, 2004.
- [3] Keiji Kieda, Hiroshi Tanaka, Tong-xiao Zhang, "On-line Optimization of Avoidance Ability for Redundant Manipulator," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*, Oct. 9-15, 2006, Beijing, China
- [4] Ze Chu, Peng Cao, Yan-Ming Shao, Dong-Hai Qian, Xiang-Can Wang, "Trajectory Planning for a Redundant Mobile Manipulator using Avoidance Manipulability," *Proc. of IEEE Int. Conf. on Automation and Logistics*, Aug, 2009, Shenyang, China.
- [5] Wang Qizhi, Xu De, "On the Kinematics Analysis and Motion Planning of the Manipulator of a Mobile Robot," *Proc. on Chinese Control and Decision Conference*, 2011.
- [6] Joel W. Burdick, "Global Kinematics for Manipulator Planning and Control," *Proc. on Signals, Systems and Computers*, vol. 2, pp. 1002-1007, 1989.
- [7] Paul J. Besl, Neil D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, No.2, pp. 239-255, Feb., 1992.
- [8] Maja Temerinac, Marco Reisert, Hans Burkhardt, "Invariant Features for Searching in Protein Fold Databases," *International Journal of Computer Mathematics*, Vol. 84, No. 5, pp 635-651.2007.
- [9] Liila Torbi, Kamal Gupta, "Integrated View and Path Planning for an Autonomous six-DOF Eye-in-hand Object Modeling System," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2010, Taipei, Taiwan.
- [10] Dugan Um, Dongsuk Ryu, MyungJoon Kal, "Multiple Intensity Differentiation for 3D Surface Reconstruction with Mono-Vision Infrared Proximity Array Sensor," *IEEE Sensors Journal*, vol. 11, no. 12, pp 3352-3358, Jun, 2011.

Chapter 8

Perceptive Parallel Processes. Coordinating Geometry and Texture

Perceptive Parallel Processes Coordinating Geometry and Texture

Marco A. Gutierrez¹, Rafael E. Banchs² and Luis F. D'Haro²

Abstract—Finding and classifying specific objects is a key part in most of the tasks autonomous systems could face. Properly being able to reach objects and find their exact location is very important for successfully achieving higher level robotic behaviors. To perform full object detection and recognition tasks in a wide environment several perception approaches need to be brought together to achieve a good performance. In this paper we present a dual parallel system for object finding in wide environments. Our system implements two main parts. One texture based approach for wide scenes, composed by a Multimodal Deep Learning Neural Network and a syntactic distribution based parser. And another specific geometry based process, using three dimensional data and geometry constrains to look for specific objects and their position within a whole scene. Both systems run in parallel and compliment each other to fulfill an object search and locate task. The major contribution of this paper consists on the success of combining texture and geometry based solutions running in parallel and sharing information in real time to allow a full generic solution to be able to find almost any present object in a wide environment. To validate our system we test it with real environment data injected into a simulated environment. We test 25 tasks in a household environment obtaining a 92% overall success rate finally delivering the correct position of the object.

I. INTRODUCTION

Significant amount of work has been done in scene understanding from 2D images since the beginnings of computer vision research, achieving significant results. Hand-designed features such as SIFT [1], ORB [2] or HOG [3] underpin many of these successful object recognition approaches. They basically capture low-level textured information with the difficulty on effectively capturing mid-level cues (like edge intersections) or high-level representation (like different object parts). Recent developments in deep learning based solutions have shown how hierarchies of features can be learned in an unsupervised manner directly from data. Learned features based solutions proved significant improvements on object recognition and detection, achieving some of them up to around 90% success rates on different benchmark training/testing sets (i.e. The Pascal VOC Challenge [4]). Recently even full semantical well structured image descriptions are generated by the latest multimodal neural language models [5]. Still when using 2D based scene understanding a lot of valuable information about the shape and geometric layout of objects is not considered. Adding

*This work was supported by the A*STAR Research Attachment Programme.

¹Marco A. Gutiérrez is with the robotics and artificial vision laboratorio (RoboLab), University of Extremadura, Spain marcog@unex.es

²Rafael E. Banchs and Luis F. D'Haro are with the HLT dept., I2R, A*STAR, Singapore. rembanchs@i2r.a-star.edu.sg, luisdhe@i2r.a-star.edu.sg



Fig. 1. Our system combines the best of 2D and 3D data information through to coordinated parallel process.

geometric information on these solutions could generally improve their results as well as enrich the information they deliver as an output.

On the other hand, 3D model based approaches make easy to reason about a variety of properties from volumes, 3D distances and local convexities. Solutions focusing on object shapes and geometric characteristics have had also intense computer vision research focus, specially due to the recent new range of inexpensive and fast RGB-D sensors available in the market. 3D features such as FPFH [6] or NARF [7] are some examples of robust features that describe the local geometry around points for 3D point cloud datasets. However 3D solutions have some drawbacks when dealing with heavily clustered scenes or very general views of the environment.

Although good solutions exist on both, image and point cloud based approaches, when it comes to solving tasks in real environments, a more generic approach to achieve a solution for the problem is needed. Systems with a use of both 2D images based solutions and 3D geometry aware processes can provide a more generic purpose robotics architecture with more reliable and rich information. Our approach combines the rich information obtained from new multimodal neural network object classification techniques on general 2D image scenes with a 3D geometric, distance and shape aware process (figure 1). This allows us to minimize the drawbacks of each of each approach with the strengths of the other.

For the evaluation of our model we used a hybrid simulation-real scenario. A simulator tool was used to man-

age the robot movements around the environment while sensor data was injected into the system from real scenario captures. This allowed us to test our approach with real environment data, since all perception information used as an input for our application comes from real sensors. As a result we obtain quite promising results on the object finding tasks tested.

The remaining of the paper is organized as follows: in section II we provide an overview of some related works. Following section III gives a detailed general description of the perception system. Section IV and V explain more specific details regarding each of the two main processes, the texture aware process and the geometry aware one respectively. Finally we evaluate the system with an experiment in section VI and give some conclusions and future lines of work in section VII.

II. RELATED WORKS

There is a wide range of research in the area of scene understanding and object recognition from 2D and point cloud data. With RGB-D increased popularity, bringing an easy to access way to RGB and depth data at the same time, several researches have tried combining the two sources of information.

Sensor fusion approaches are the most common ones, they take both sources of data and combine them into one system to improve performance. I.e. in [8] they associate groups of pixels with 3D points into multimodal regions that they call regionlets, then they measure the structure of each regionlet using bottom-up cues from image and range features. This way they are able to determine the scene structure separating it into the meaningful parts discarding the background clutter. Although they do not rely on any rigid assumptions about the scene like we do (we consider objects are placed on tables), the output provides a basic structure discovery over a scene with detection of the main objects while our solution solves a specific object search and locate task on a wider environment.

The machine learning based approaches take features from both depth and color data sources and combine them into one multimodal space to perform later searches for a given input. Koppula et al. [9] perform a labeling task on over-segmented 3D RGBDSLAM sensed scenario. They build a graphical model capturing 2D images information (local color, texture, gradients of interests, etc.) as well as local shape and geometry, and geometrical context (where object most commonly lay to each other). This model then uses approximate inference and is trained using a maximum-margin learning approach. They show the benefits of using image and shape against separated solutions. Also, Lai et al [10] present an RGB-D Object Dataset and evaluate some object recognition and detection techniques. They combine 2D SIFT descriptors with efficient match kernel (EMK) features computed over spin images on randomly subsamples set of 3D points. These features are then used for the evaluation of three classifiers: a linear support vector machine (LinSVM), a gaussian kernel support vector machine (kSVM) and a

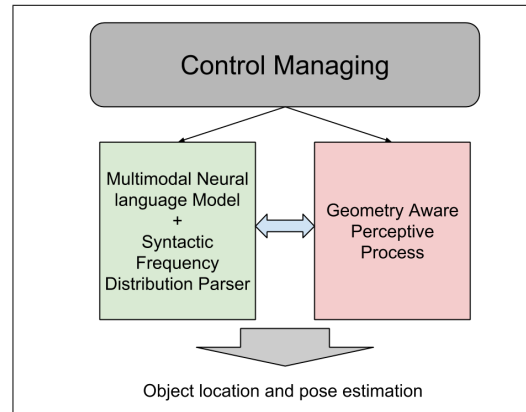


Fig. 2. Overview of the architecture of the perception system.

random forest (RF). The main difference with these works is that they restrict the search to a certain scene while our solution provides a framework to solve a find and locate an object in an entire household environment.

The work on [11] combine high-resolution 3D laser scans with 2D images to improve object detection. Their solution relies in using a sliding window approach over a combination of visual and depth channels and use those patches to train a classifier. It solves the same problem as the one presented here although they do not perform any optimization in terms of the path to reach the object most probably leading to a slower solution for an object search and location like the one explained here.

Also in [12] they use binary logistic classifiers on 2D and 3D features. The 2D features are small patches selected from images on a training set. They, then, compute 3D features from distance from robot estimation, surface variation and orientation and object dimensions. These features are then learned by the classifier over two-split decision for each object class. The difference with our solution is that they learn multimodal models per object while here the rgb and point cloud data are used by two different process and the outcome combined in a final solution.

III. THE PERCEPTION SYSTEM

As shown in figure 2, the system's architecture has a control manager for decisions and mediation among two perception parallel process. This manager takes care of the information shared between both processes and delivers notifications according to them.

The *texture aware perceptive process* (showed in figure 2 in green) exploits 2D images information data. It runs a multimodal neural language model as described in [13] along with a syntactic frequency distribution based parser to process and evaluate the neural network output. The second one, the *geometry aware perceptive process* is exploiting the geometric features of the environment. This one takes care of two main tasks, looking for tables through the point cloud data and segmenting tabletop setups, recognizing the object

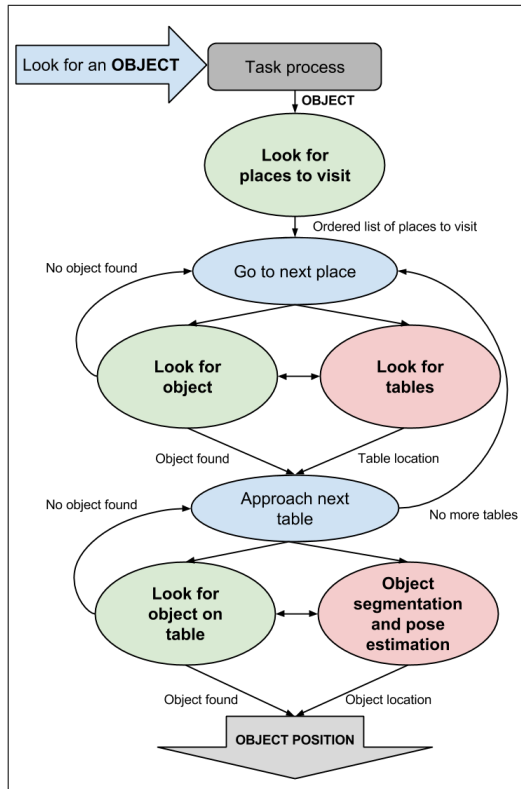


Fig. 3. Flow of the different states and process on the system.

and estimating its position through a shape and position aware histogram based feature matching system.

Figure 3 describes the states and process of the system for a certain object search and locate task. Green tasks are performed by the *texture aware perceptive process* while the ones in red belong to the *geometry aware perceptive process*. A restriction the system assumes is that objects are placed on tables. A simple task is given to the robot in the form of “Look for the OBJECT”, and the object name is extracted and passed to the *texture aware perceptive process* for the “look for places to visit” step. A list of generic images for each available place is stored in our database and evaluated by this perceptive process. A frequency of appearance of possible objects histogram is built for each place. Places to visit are then ordered according to highest appearance of label of the object on this output. Places with no object appearances are left to visit last and ordered randomly.

Once the list of places to visit is ready, the robot visits them in order. When the first place is reached both processes start to work in parallel for the required object. The *texture aware perceptive process* provides a frequency distribution of objects on images taken from the current place while the *geometry aware* one will start looking for tables on the scene point cloud data. If the object is found in a scene image and a table has been detected the robot will start moving towards

the table. Once a table is reached the *texture aware perceptive process* keeps validating the appearance of this object in the scene, then a tabletop segmentation process will be started by the *geometry aware perceptive process* in order to segment, recognize and locate the object.

If no object seems to be present when the tabletop segmentation is performed, the robot continues with the next table or with the next place in list if no more tables are available in the current place. We will only conclude that we cannot find an object once all places have been visited and no object has been found.

IV. TEXTURE AWARE PERCEPTIVE PROCESS

This texture based perceptive process is intended to get quick scene labeling from wide overviews of the environment. It contains a previously trained multimodal neural model that outputs image descriptions. Then, taking into account the top nearest descriptions in the model, a parser extracts the object candidates and builds a frequency distribution histogram on the appearances of these objects class names. This frequency distribution histogram helps obtain a more robust output against false positives as the objects that are present in the scene tend to keep appearing with higher frequency over time in the sentences while the false positives have usually a much lower frequency.

A. Multimodal neural model

As previously mentioned the multimodal neural model follows the structure in [13]. This is a neural model pipeline that learns multimodal representations of images and text. The pipeline uses a long short-term memory [14] (LSTM) recurrent neural network for encoding sentences. We use a convolutional network architecture provided by the Toronto Convnet [15] in order to extract 4096 dimensional image features for the neural model. These image features are then projected into the embedding space of the LSTM hidden states. A pairwise ranking loss is minimized in order to learn to rank images and their descriptions. For decoding, the structure-content neural language model (SC-NLM) disentangles the structure of a sentence to its content, conditioned on distributed representations produced by the encoder. Finally, the output is generated by sampling from the SC-NLM the image top descriptions.

B. Syntactic frequency distribution parser

After the system obtains the top scenes generated descriptions, it extracts potential object classes from them, using a syntactic parser. Using the Neural Language Toolkit [16] we syntactically analyze the sentences to extract object candidates that could be present in the image. A frequency distribution histogram is computed over this object candidates. This histogram is then used to evaluate the believe that an object is present in a scene, allowing us to compare different scenes according to the probability of finding an object there and therefore discriminate possible false positives.

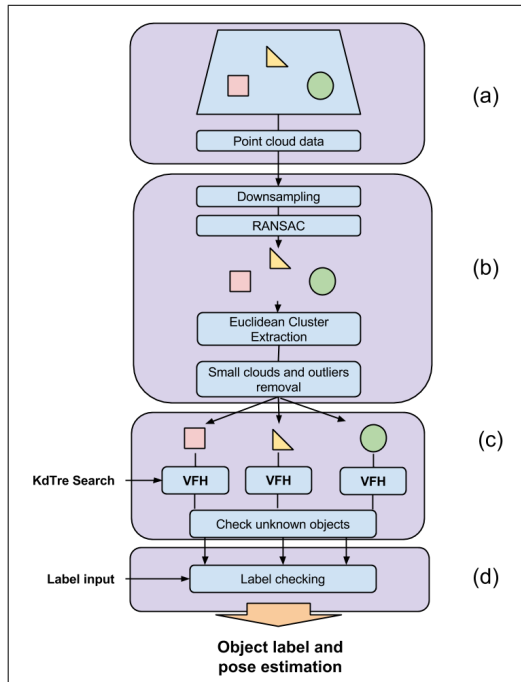


Fig. 4. Tabletop segmentation and object recognition pipeline using point cloud data.

V. GEOMETRY AWARE PERCEPTIVE PROCESS

This process exploits the geometry present in the environment to extract a wide variety of information. For our approach we have restricted the task of finding objects, to objects placed on top of tables. Therefore this process performs two main tasks, one is looking for tables in broad scenes and another one consists on a tabletop segmentation with shape based object recognition and pose estimation.

A. Looking for tables

We describe tables as planes that are parallel to the floor and found at a height between 40 and 110 centimeters. Therefore, we use the RANDOM SAMPLE CONSENSUS (RANSAC) [17] for plane model fitting in the scene point cloud data with a previous downsample of 1cm. Using this algorithms we recursively look for planes matching the previously mentioned constrains and label them as tables.

B. Object recognition and pose estimation

The tabletop segmentation is used when a table is approached and in order to recognize the objects on top of it as well as to estimate their final position.

In the first part, shown in figure 4.b, the RANSAC algorithm provides us with the plane equation and the points that match that equation. Since the RANSAC uses a threshold to deal with sensor noise, points matching the model are not in a perfect plane but within a certain range, so we first project this points to fit the plane equation to obtain a perfect

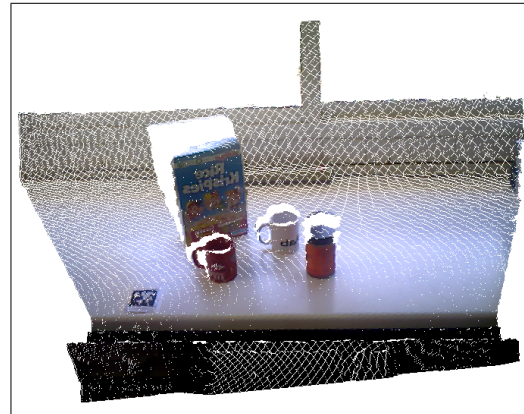


Fig. 5. Example of one of the tabletop setup used in the experiment.

plane point cloud. Then we obtain the convex hull of these plane point cloud and perform a bounding box on top of it up to a certain high. Points within the bounding box are then considered to correspond to objects sitting on top the table. Then it is performed an euclidean clustering extraction to segment the object candidates point clouds.

As the next step (figure 4.c) we compute these point clouds Viewpoint Feature Histograms [18] (VFH) and look for the nearest match in our database. For this database we have a previously computed VFH of single views of objects. These VFHs are stored and retrieved through fast approximate K-Nearest Neighbors (KNN) searches using kd-trees [19]. The construction of the tree and the search of the nearest neighbors places an equal weight on each histogram bin in the VFH and spin images features.

Finally the system would check if any of the labels from the objects correspond to the one we are looking for, see figure 4.d, and call it a success or not.






VI. EXPERIMENT

We perform several experiments sending the robot to retrieve different objects in a wide household environment. For the experiment an hybrid simulator-real data environment has been used. We used the simulator for the robot movements between places, while sensor data has been acquired with real RGB and RGB-D cameras (i.e. the tabletop showed in figure 5) and matched to the specific locations on the virtual plane. When the robot needs to move around the simulator takes care of it, once certain positions in the map are reached, the previously obtained real data is injected and used as input for the algorithms. The robot always starts at the entrance of the apartment and from there performs the most optimal way to find the object and delivers its estimated position as a final result.

A. System setup

The LSTM encoder and SC-NLM decoder from the multimodal neural model have been trained using a combination of the Flickr30k [20] dataset and the Microsoft COCO

TABLE I
SUCCESS RATES ON THE DIFFERENT PARTS OF THE ALGORITHM

OBJECT	1. Places to visit Ordering	2. False Negative	3. False Positive	4. Success Rate
Cereal box 	5	0	0	100%
Cup 	5	0	1	80%
Bottle 	5	0	1	80%
Laptop 	5	0	0	100%
Monitor 	5	0	0	100%
Overall rate	100%	0%	8%	92%

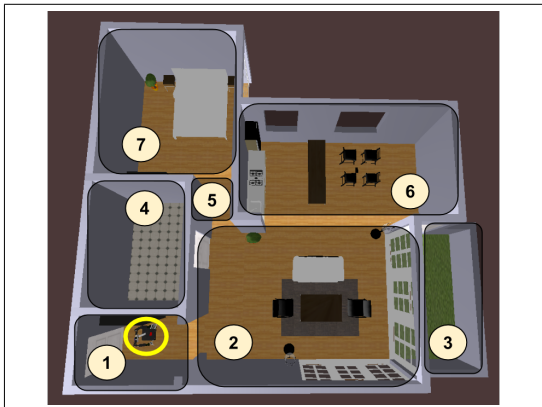


Fig. 6. An overview of the simulation household environment. The rooms are labeled as follows: 1.- Entrance, 2.- Living room, 3.- Patio, 4.- Bathroom 5.- Hallway, 6.- Kitchen, 7.- Bedroom. Circled in yellow is the robot at its starting point.

dataset [21]. The 4096 dimensional image features for the multimodal neural model training are extracted using the Toronto Convnet with their provided models. The frequency histogram is built using the NLTK toolbox on the top 5 generated sentences over at least 5 frames, to achieve a robustness on the objects observed. This NLTK tagging and syntactic analysis is performed using the Treebank Part of Speech Tagger (Maximum entropy) they have available. For the rooms representation, images in the house 5 generic different images of parts of a house are used for each of the places in the house: entrance, room, kitchen, living room, bathroom, patio and bedroom. This images have been selected so they contain the usual set of items presents in

those rooms. For the point cloud analysis a kd-tree stores 3729 VFH from different views of 75 different objects.

All the system is developed using the RoboComp robotics framework [22] and the simulation is performed in a virtual scenario using the RoboComp simulator tool. See figure 6 for an overview of the simulation environment.

B. Results on the experiments

We run 5 different tasks 5 times and collect the results in the table I. First we measure if the ordering of places to visit after the “Look for places to visit” step in our system was optimal (check figure 3 for details). This turned out to work perfect for all of our test cases, basically because some of the description pictures of the places contained those items and the *texture aware perceptive process* was able to detect them. It is important for this step to select a good range of images representing the different places to visit (see figure 6), specially those images that clearly show an average of the objects you can usually find in those places.

Then we count the false negatives occurrences, this is when we are done with the searching and no object was found. Along our testing this never happened and an object was always found. However we obtained two false positives when the system mistaken a cup for a bottle and when a bottle was mistaken for a bottle of glue. Those mistakes are basically due to the similarity on these objects shape. We could avoid this in the future reinforcing this step with other object features. Specially since the objects to be found where actually present in the table being segmented at the time.

The final success rate on obtaining the proper location of the object and pose estimation is quite high which results promising for further real applications of the system.

VII. CONCLUSIONS AND FUTURE WORK

We presented a hybrid perception system that combines 2D data based solutions and approaches using point clouds running in parallel and sharing information in real time in order to achieve a finding object task. The system is able to successfully predict a route through the places with higher probability of finding this objects. We obtained a high rate of success in our experiments as we only obtained two false positives among all our test cases.

An interesting future work would be to perform further testings with a wider range of objects. This could help find some weak points on the system that we might have not found yet and that should be worth to strength with more processes interaction. In the same line and although the sensor data used in the testing were taken from real sensors, integrating the solution with a real robot could bring a more accurate overview of how the system performs in real environments.

False positives obtained during experiments are mainly because of a bad performance of the *geometry aware perceptive process*. Since similarity on the shape of different objects confuses the VFH search, exploiting texture based features on this last step could most probably benefit the whole system final output. Also, since we are using an euclidean clustering extraction method for objects on top of the table, our system cannot deal with heavy cluttered scenes or objects touching each other. Adding alternatives to the segmentation process could help improve this in order to cover a more varied range of scenarios. It would be also desirable to avoid the assumption that objects are always on tables, so we should look into new ways of scene segmentation to improve this step.

Finally adding a learning process in the system would be an interesting enhancement, both parallel process could complement each other, correcting each other mistakes and providing the fixed mistake as a new source of learning, leading to improvements in the following overall system performances.

REFERENCES

- [1] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999.
- [2] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision, ICCV '11*, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.
- [3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.
- [4] Mark Everingham, S.M.Ali Eslami, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.
- [5] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.
- [6] R.B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3212–3217, May 2009.
- [7] Bastian Steder, Radu Bogdan, Rusu Kurt, and Konolige Wolfram Burgard. Narf: 3d range image features for object recognition. In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics*, Int. Conf. on Intelligent Robots and Systems, IROS '11. IEEE Computer Society, 2010.
- [8] Alvaro Collet, Siddhartha S. Srinivasa, and Martial Hebert. Structure discovery in multi-modal data: A region-based approach. In *ICRA*, pages 5695–5702. IEEE, 2011.
- [9] Hema S Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Advances in Neural Information Processing Systems*, pages 244–252, 2011.
- [10] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.
- [11] M. Quigley, Siddharth Batra, S. Gould, E. Klingbeil, Quoc Le, Ashley Wellman, and A.Y. Ng. High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2816–2822, May 2009.
- [12] Stephen Gould, Paul Baumstarck, Morgan Quigley, Andrew Y. Ng, and Daphne Koller. Integrating Visual and Range Data for Robotic Object Detection. In *ECCV workshop on Multi-camera and Multimodal Sensor Fusion Algorithms and Applications (M2SFA2)*, 2008.
- [13] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.
- [14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
- [15] Toronto University. Convolutional Neural Nets. <https://torontodeeplearning.github.io/convnet/>, 2015. [Online; accessed 04-March-2015].
- [16] Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the COLING/ACL on Interactive Presentation Sessions, COLING-ACL '06*, pages 69–72, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.
- [17] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.
- [18] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162, Oct 2010.
- [19] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.
- [20] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.
- [21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [22] P. Bustos Marco A. Gutiérrez, A. Romero-Garcés and J. Martínez. Progress in robocomp. *Journal of Physical Agents*, 7(1), 2013.

Chapter 9

Integrating Planning Perception and Action for Informed Object Search

Part II

Conclusions and Future Works

Chapter 10

Review and contributions

Throughout this thesis a system for object informed search using different planned perception processes was researched. The different steps were carefully developed and tested to form a complete system able to make a real robot deliver objects in a large household scenarios. Several techniques were developed for each one of these steps that enhance current state of the art and boost the performance of the robot in a complete search and find task. The goal of this system is to make the robot able to fulfill requests from a human asking it to bring specific objects in a large real environment. The contributions of this thesis were made, in their majority, with a real robot in an apartment-like environment (the Shelly robot). The idea was to provide robust solutions for each and everyone of the stages of the perception system.

Most of the perception processes that were used in the system have a cognitive connotation. The system makes use of internal information of the robot in combination with the sensed data when a search is performed in order to enhance the results. This is called informed search, as it is an aided search that not only uses the data obtained by the sensors in the specific moment of the search. Consequently, in this work, when the robot has to search for an object, it takes advantage of its extra cognitive information. The robot combines information from its own internal knowledge, acquired along its lifetime, with the information obtained from its sensors. The use of prior information –the robot’s internal world model, which includes knowledge about objects– is considered in every search and allows the robot to reduce search times.

The different steps of the cognitive perception were planned accordingly to the goal of the task, the cognitive knowledge in the robot and the sensed data from the environment. An internal world model in the robot was used to maintain an internal state of the surroundings, while some extra information regarding perceptions was also stored through the form of embeddings. Planning was made through the internal model but also taking into account information from the embeddings and other extra sources such as sensor information. This helped maintain a flexible set of stages that robustly guide the high level tasks of the robot, making it able to react to unexpected situations and to deliver the requested item.

After the object is detected and reached, the exact location, along with some extra information, is estimated for the final grasping. The object location is determined to let the robot know where exactly is the target object. Some techniques were developed to improve the information obtained from the object, such as volume or shape, in order to perform a final grasping movement. Finally, a SPAM technique was proposed to help the robot arm find its way to the target to grasp it and deliver it to the human.

Along the research conveyed in this work, several contributions to open source robotics and

perception projects took place. These contributions not only help improve the state of the art, but, in the form of open source tools, help the research community allowing them to reuse the work from this dissertation. These projects constitute an essential means for research, easing the reuse of the algorithms, their improvement, deployment or even the development of new ones. Most of the source code of these tools available through open source repositories along with instructions and manuals for easy understanding.

The major contributions in software were made to RoboComp, as it was the main robotic framework in almost all the works involved in this dissertation. Contributions were made to its model-oriented design which helps building components through different Domain Specific Languages (DSL). Efforts were also made in its set of libraries, tools and components that help improve the robotics development cycle, specially on those related to perception.

OpenDetection was developed and improved as part of the work in this dissertation. It aims to unify perception algorithms into one common interface to ease their use by roboticists. It offers frame generators, trainers, detectors and detections. Frame generators are used to grab the data from camera sensors (either RGB or RGBD). Trainers are used for those algorithms that need training. Detectors are the ones performing the actual detection and the ones that will produce, as an output, a certain detection. With this library a scene can now be easily processed through different (trained or not) detectors and producing different detections through a common API. All detections were structured in a similar way so they can be easy to compare and or combine for enhanced results. In OpenDetection, algorithms for both 3D and 2D data were integrated. It was designed to be easy to use and to reduce code complexity.

Some other minor contributions to robotics software were also developed and published. A data grabber for IPA sensors [Um et al., 2011] was developed and integrated as part of the Point Cloud Library. A perception pipeline was developed for a KUKA OmniRob [Shepherd and Buchstab, 2014]. Also some other perception developments were created for the RoCKin robot challenge and are now integrated in the perception component layer of RoboComp.

Chapter 11

Future works

Although a full setup of a delivery robot in a household environment was successfully achieved with this dissertation, further work is still needed if we want to bring these solutions to real world scenarios where constant changes are common. Even though through the real delivery robot was deployed and tested in a real apartment, further exhaustive testing and debugging is needed to make this application robust enough for a day to day usage. Some small and specific autonomous robotic applications might be able to make it now to a final market product, however, for most of the real high level ones there are still a lot of issues to improve.

Sensing is a field that has seen a lot of improvement in the last few years, specially with the mass production of PrimeSense devices and the price drop of Velodyne 3D LIDAR sensors after research efforts pushed by programmes such as the DARPA Challenge. However, further research in this area is still needed as the sensors currently available are still not perfect. As an example, PrimeSense devices have an average of 1 to 2 cm. of noise and LIDAR sensors don't work properly under rainy conditions or against some materials such as glass. These drawbacks produce errors that propagate and make more difficult the whole perception process for an autonomous robot.

Regarding the room modeling work presented here, a clear improvement would be the relaxation of the rectangle assumption. Although this assumption is true for most of the rooms in indoor environments it is not true for all of them. Other structures can also appear when working indoors such as corridors or stairs.

Selection of possible initial search spaces needs of prior information in the robot. Further work using Internet based information retrieved through search engines or shared databases (*e.g.*, the RoboEarth project) can help improve this informed search step of the robot. Other semantic relationship tools can also be tested in order to improve the reasoning on the semantic information that different labels provide.

Even though the object detection, recognition and location techniques presented here worked well for our delivery robot use case, they are hard to be directly extrapolated to other high level tasks, specially if the objects used are completely different. These methods should be expanded to cover more and different objects. They can also be improved to increase the success rate. Cognitive interaction with the object detection process should be increased in order to help the robot reason about the scene and reduce the possibility of failures.

Expanding the algorithms regarding shape retrieval can help cover more complex object forms. It would be desirable to be able provide shape information on asymmetrical objects to the grasping stage. Fitting generic meshes into objects shapes or breaking non symmetrical

objects into their symmetrical parts could be a starting point in this direction. Extending the use of cognitive information in this step can also be helpful to decide different approaches that might improve the shape fitting process. Other cues, such as curvature and texture may also be helpful in predicting the complete shape for single viewed objects.

The whole integration of the system can also be improved increasing the world details included in internal model for a more accurate planning. Attention mechanisms can be added to the system in order to make the robot actively look for objects while moving. This can improve the speed of finding objects or can even enable the robot to look for more than one object at a time. Finally, other and higher level tasks are also desirable, therefore integrating this “object delivery task” into higher level ones would be a good improvement towards more intelligent and autonomous robots.

Appendix A

Robotic Software Contributions

“There’s nothing sadder than a puppet without a ghost, especially the kind with red blood running through them.”

— Batô, Ghost in the Shell

A.1 The Needs of Specific Software for Robotics

The development of a complete autonomous robot is a very hard task that involves a challenging intersection of numerous software and hardware modules. It comprises a wide range of disciplines that ranges from Electronics, Artificial Intelligence, Mechatronics, Software Development, Computer Vision all the way to areas like Psychology, Neuroscience or even Philosophy. The problems that are usually faced when carrying out the development of a complete robot easily surpass the possible efforts of a single roboticist. Therefore development and integration is usually accomplished by several people and more than usually involving different teams from different research institutes. On top of that, the reuse of software is often needed to achieve a full autonomous system in order to integrate state of the art developments from the different fields involved in a robot’s development.

Software complexity, from the developer point of view, is an important issue because scalability decreases as complexity increases. When trying to implement heterogeneous robotics solutions the complexity of the software involved increases by orders of magnitude as we add new functionalities to the system. Dealing with this complexity aims for specific deploying, management and testing tools that would help a roboticist achieve a smooth integration process.

Because of the nature and purpose of autonomous robotics solutions they are doomed to share their environment of action with human beings. Robustness becomes a key asset in these situations as unexpected or failure robot actions may not only change the outcome of a designated task but end up with devastating consequences such as damages to the environment materials or even harm to surrounding humans. Achieving robustness is not easy, specially when dealing with state of the art solutions and the mentioned complexity that these systems bring. Tools that can help perform heavy tests on the software and hardware modules are needed, like simulated environments where the software can be tested without real harm to surrounding when unexpected situations happen.



Figure A.1: The third generation of the autonomous robot Shelly entirely constructed and developed at Robolab at the RoCKin Challenge 2014 in Toulouse, France.

Due to the big challenges that research on these fields arise more than often a single robotists research focuses on one of these fields. In the end different software modules that are needed for the integration of one robot come in different programming languages or even targeting different platforms. On top of that robots usually do not only have a unique computer architecture but they integrate different types of hardware like from regular x86/ARM64 personal computers , ARM based solutions (Like Raspberry Pi, Odroid, Cell Phones, etc.), arduinos or even microcontrollers and specific hardware. Problems also arise when integrating third party hardware in a robotic platform. Vendors may only provide support on certain platforms (The Intel Real Sense platform or Microsoft Kinect 2 drivers are initially only provided for Windows Operating Systems), therefore support for software modules written in different languages and running on different platforms must be provided by the relying software.

Since the number of different architectures mentioned and the continuous upgrades a robot suffers, hardware independence becomes a desirable feature. Minor changes to configuration should be required to adapt existing software and allow reuse when changing the underlying hardware. The Hardware Abstraction Layers that are usually included in the different robotics frameworks are developed with the goal to achieve this hardware independence and allow the reuse of software modules among different robots or when changes to hardware occur.

Since the high complexity of the software and the amount of different architectures a robot holds it is also desirable to be able to distribute the execution of this software. Being able to distribute the software helps balance the high CPU and memory load involved algorithms use. Developers can even choose different architectures that better fit the different algorithms like using GPU clusters for algorithms that are intensive on matrix operations, i.e. computer vision algorithms.

As stated here, robotics development has a very wide range of specific needs that need to be addressed. Robot software developers have commonly agreed the need of usage of components oriented programming in order to be able to decompose the problem of a robot development into smaller parts. This way these modules can be reused, distributed (to achieve Robotics frame-

works have become quite popular among roboticists as they provide the means to support these components development and testing with the necessary tools and libraries. A full overview and comparison of the main current robotic frameworks can be found on Section A.4.

A.2 Component Oriented Programming

This section introduces the basics of component-oriented programming (COP) [He et al., 2005], a programming paradigm very important in the robotics field. COP enables software solutions to be constructed through prebuilt software components, that are reusable, self-contained blocks of computer code. These components need to follow certain predefined standards including interface, connections, versioning, and deployment. Every one of the components in the solution should be reusable and independent of context. In COP interfaces and composition are emphasized. In this sense, we could say that COP is an interface-based programming. Clients in COP do not need any knowledge of how a component implements its interfaces. As long as interfaces remain unchanged, clients are not affected by changes in interface implementations.

Although initial works on COP are relatively old [McIlroy, 1969], its widely used in almost any software that is designed for robotics. Almost any robot runs COP based software, anything ranging from autonomous manipulators, assistive robots, mobile platforms, social robots up to even small education oriented platforms. The main reason for this wide spreading of COP in robotics is that it offers developers an easy way to create scalable, flexible and reusable software, key features for robotics software developers.

A.2.1 Main Characteristics

The idea behind component-oriented programming (COP) is designing and developing complex systems as networks of standalone modules, so that the goal of the whole system is achieved through the interaction among the elements of the network. These independent programs are called components, and the software layer they use to communicate is usually called middleware. Each component has a component interface which other components can use to interact with it. When using component interfaces they can be seen as the API that other components can use to communicate with each other, much like C++ class definitions. In fact, as virtual classes, component interfaces can be implemented by more than one component (*e.g.*, the same interface can be provided by several components).

The main purpose of COP is to develop software by assembling components that exchange some sort of information. The definition of component might be quite blurry and overloaded, making the term confusing in current software engineering. In different contexts a component may refer to different things, however in robotics its used to refer to integral entities that perform some specific work or service. Components can act as servers, clients or both depending on the system state. In the case of robotics all the components usually have a similar structure that helps build the communication and other common interfaces, such as testing. Then only the specific behavior in the component is added as a behavioral differentiation among them.

As said components can communicate with each other through different communication patterns [Schlegel, 2006]. For this components behave as either service requestors or service providers at a certain point. Most of them act as both depending on the needs of the system. Two regular communication patterns are commonly used among roboticists. A synchronous one

where the client requests the information from the server usually in the form of a function call such as Remote Procedure Calls (RPC). And an asynchronous one where the server publishes the information and clients get called once this information is available, this one is usually referred as publication/subscription. Robotics frameworks use these patterns to communicate their components. It does not usually make a huge difference to choose one or another, both have their advantages and disadvantages. Lately most of the frameworks support both and users can choose whatever they prefer for each component to communicate.

A.2.2 Advantages and disadvantages

One of the main advantages of using components for development is the possibility of reusing the same components on and on again, as stated in [McIlroy et al., 1968]. Since these components are individual units that perform very specific tasks, they can be reused in different occasions that the task is needed. *e.g.*, A component in charge of obtaining images from a camera and exposing them to the rest of the system can be reused anytime images from the camera are needed. In robotics this is a key advantage since building a whole robot implies a strong effort an expertise usually taking several years of development. Having key reusable parts from one robot to the other or even from one developer group to the rest helps mitigate this huge amount of work.

Another key feature that heavily helps robotics on their task is the ability of balancing the load of the algorithms. Having different units providing services that run as separate components helps the operating system schedule and parallelize these processes, which then translates into a more efficient usage of the computer resources. On top of that, due to the distributed aspect of COP, components can be deployed in different machines provided that they are interconnected in some way or another. This way the load of the components can be distributed among different machines with different capabilities that matches the needs of the algorithms. This is extremely helpful for robotics development since some of the algorithms like the ones used for the robot's cognitive vision make a heavy use of memory and CPU. Therefore these component that need lots of resources can be isolated in their own machines or even in specific architectures for their tasks such as GPU clusters or high performance computers.

Decoupling the software into small modules that take care of specific purposes also brings benefits to future code maintenance tasks. Apart from the mentioned reusability code also gains in readability, since there is a somewhat a predefined structure that software must follow. Having small units of functionalities also helps with the testing. Since these units usually have a very specific and predefined purpose they can be easily tested with a set of inputs and desired outputs. However in the robotics field individual testing of the components does not mean a good performance of the overall system. Specially due to the fact that usually lots of these components are needed for a single task and that they also depend on the environment conditions final behavior is not guaranteed and full system testing in real environments is always required.

Usually, although it might depend on the tools used, components can most of the time be developed in a wide range of languages. This means developers can use the full qualities of a certain language for one component and then use another language that might match the algorithm needs better for another component, taking full advantage of the availability of languages and even their associated libraries. It also helps developers to easily port and use software written in different languages. This is very important as the reuse of software for roboticists is crucial. Sometimes certain work is the product of one or two full PhDs and it might be written

in a certain language different from the usual one used in the robot system. Porting the work to a new language might require intensive work, however being able to import it as a component in its own language can save roboticists lots of time. This way the roboticist only has to add the code to a certain component and communicate with it through the network to obtain the algorithm output.

In the same line they can be compiled and run in different Operating Systems (OS). This allows the developer to have different machines with different OS. Although robotics is mostly developed under GNU/Linux based systems as a norm, sometimes there are cases that require software to be run in other OS such as the Shunk motor drivers which have their latest drivers only available for Windows [sch, 2016] or the Microsoft Kinect 2 which only gives support for a specific OS [kin, 2016].

One of the drawbacks of the COP is that it puts a certain load on the network. Specially when the data exchanged among components is considerably big this load can become quite big and the frequency of update of the client components gets penalized by the throughput available in the network. This is actually a common issue in robotics, since some of the updates might be critical, *e.g.*, a late update on the localization system might cause the whole robot to crash into some obstacle. Therefore it is encouraged to try to use networks with higher bandwidth and try to minimize the data that goes through low bandwidth ones such as a WiFi network.

However there are also disadvantages when you run a distributed system of components. One of the main drawbacks are the communication parameters. A component will need the information on who it needs to communicate with. This information is usually provided in the form of configuration files, which adds some extra work and maintenance over the actual development. Moreover this configuration more than often differs from one deployment to another, where you might need information such as ports, addresses and/or interface names. Maintaining this configurations is a problem that still takes time on robotics deployments and has most probably has no easy solution.

It is also worth mention that even though COP is meant for the development of high reusable code and even though it usually works that way this might not be always the case. A problem that often arises and very specifically in the robotics field is the hardware dependence. Low level components that are meant to directly manage hardware pieces, which are very common in robotics, are not as reusable as one might desire. More than often these components are tied to a specific piece of hardware without which the component becomes useless. Therefore in the case of hardware updates development of new software to access it is required, reducing the reusability of the components of the system.

A.2.3 Why Component Oriented Programming for Robots?

There is a high level of complexity when developing autonomous robots that extends to hardware and software design. Due to the hardware differences among robotics platforms specific configuration and development might be needed for each of them. Also this complexity makes robotics to require software with characteristics such as easy ways for heavy testing, code reuse, scalability, distribution, hardware independence, concurrency or support for different languages and platforms. In order to suffice the needs of roboticists the proper software engineering techniques and the specific tools should be provided.

One might argue that most of the robotics modules are usually composed of image and

point cloud processing, linear algebra or machine learning algorithms that can be packaged as dynamic libraries. Actually a lot of them are currently packaged as so for their reuse not only in the robotics field. However the development of monolithic software that uses tons of libraries will not be a good solution as software for a robot. Delivering all the reusable code of a robot in the form of libraries will actually bring several issues. *e.g.*, One of them is that having different versions of the same library for testing purposes will not only mean different compilation processes but also will bring complexity to the compilation structure of the system as it might cause symbol collisions if not properly configured. Since the field of robotics is mostly in an early research state often changes are quite common and having to deal with different versions of code is occurs almost every day. Therefore having a decomposed system with modules in charge of very specific tasks that solve small problems help with code mantainance and reusability

One big problem in robotics is the software load, which more than often becomes very high, specially when running stochastic algorithms through big search spaces. The recent and extended usage of Deep Neural Networks (DNNs) [Goodfellow et al., 2016] also increased the need for faster and more specific hardware that matches application needs. On top of that the amount of computation available in a robot is limited and is usually desirable to be stuck to a minimum in order to reduce the weight load of the robot. Therefore having the ability of distributing the load of the algorithms by executing the different components in different machines, even sometimes with specific hardware that helps speed up their execution is a huge advantage.

Another hard-to-obtain feature is hardware independence. Although it might not always be achievable, the use of small modules in developed software to manage each piece of hardware from the robot helps with this independence. This allows the software developer to reuse this component everywhere the same hardware piece is being used, independent of the rest of the system. *i.e.*, this would allow the roboticist to change the cameras of the robot or the platform for new ones with the same capabilities as the old ones without having to modify the software. Due to the importance of this abstraction a “Player Abstract Device Interface (PADI)” was presented in [Vaughan et al., 2003] for the Player/Stage project [Gerkey et al., 2003]. In this work a set of interfaces are defined in order to communicate sensors with actuators through standarized APIs. However defining these interfaces might not be an easy task when trying to achieve hardware independence. Similar robotic parts might provide or need different parameters which should be included in these interfaces. If all of these parameters are added to the interface to make it usable by different hardware pieces the complexity might make it hard to use and understand. On the other hand, if too few are added the number of robotic parts it can support might be to small. *i.e.*, an RGB camera will produce different data than an RGB-D camera, in this case the software developer has to decide between two interfaces with different parameters or to consider both cases in a single one making it a bit more complex. Therefore a trade of between complexity and abstraction should be achieved, since one of the main features of hardware abstraction is that similar hardware is accessed through the same interface. It is also arguable that libraries can give a similar independence from hardware, however it is hard to maintain different versions of one library in a system. On the contrary having this small modules makes it easy to have different versions of the same software that can match perfectly to the specifics of different hardware updates. A desirable hardware abstraction would mean that once changing or updating hardware in the system, changes in the software should be minimal, like changing components or some configuration parameters.

Each programming language has its own capabilities, not only due to the intrinsic specifications of the language but also due to the libraries developed for them. Libraries are a huge help when it comes to code reusability and using the best and most advanced ones is often critical for robotics development. These libraries are sometimes only available for certain languages restricting the choices for the component developer. This also can be applied to platforms, specially when software support to certain hardware parts is restricted to those platforms. People are also a variable to take into account here as different developers might have different preferences, in terms of platforms or languages. On top of that there are specific platforms such as the recently developed NVIDIA Jetson TX1 [jet, 2016]) that might be able to speed up specific operations such as an R-CNN Multi-Object Detector [Mhalla et al., 2016]. Being able to execute and combine code on a wide range of these languages and platforms is also a desirable feature. Also depending on the system a web interface or specific cell devices might be required for remote interaction, making it essential that different modules interact remotely independently of their platform and language. The communications middleware in charge of communicating the components of the developed software is the one in charge of these support. It should be able to translate the interface calls into the specific component platform/language call. The more languages and platforms supported by the used middleware the better for the developers and the heterogeneity of the developed systems.

In order to be able to make software able to provide this features in such a complex and heterogeneous distributed system communications should also be provided. Since modules perform different tasks while interacting with each other, the middleware in charge of their communications should also be able to deal with the different synchronization problems. Most of the time, messages passed between modules, specially if the communications middleware is platform and language independent, must be serialized, *e.g.*, to convert between different data structures available in different languages. The opposite process would be executed on the reception module, although the new representation might be different and specific to the receptor language or platform.

All the mentioned aspects are desirable for robotics development. Software engineering techniques should be applied to different frameworks in order to make the full development cycle of robots as efficient as possible. Due to the wide range of problems and huge expertise that involves robotics development, people with different skills will be involved in this work. Therefore it should never be assumed the developer will have high level of software engineering knowledge. Having tools to address these experts needs and making them able to contribute to the robotics development is a key task of the robotics software. Configuring robot deployments, testing modules and solving their problems are more than often a task that involves more time than even programming the solution itself. Therefore software that, along with improving the reusability, reduces the time involved in development (including configuration for deployment, testing and problem solving tasks) would be a major achievement towards improving software for robots.

A.3 Model Driven Engineering for Robotics

Model Driven Engineering (MDE) is a generic methodology that exploits domain models in order to create generic models, which are conceptual models of all the topics related to a specific problem. This enables software developers to work in higher levels of abstraction in order

which increases their efficiency. It also helps users that are not familiar with general-purpose programming languages by abstracting the implementation and making development easier and simpler. Through the use of MDE standardized models can be reused between systems, reducing development time and efforts. Since MDE uses models of recurring design patterns within the application domain, it helps easing the whole process of design. Additionally due to the standardization of the terminology in the application domain promotes and helps with communication between developers, different teams and users of the application domain. If a modeling paradigm for MDE is properly designed, a user familiar with the domain should be able to recognize the models and they should be able to be used for implementing systems.

There are different MDE initiatives, in particular, it is worth noting the OMGs Model-Driven Architecture (MDA) [Brown, 2004]. This methodology keeps the system specification (model) separated from the system implementation. MDA models are structured by layers with different levels of abstraction. With this structure, platform-independent models (PIM) can be built which provide high-level designs, and platform-specific models (PSM) which contain those elements that depend on the final system implementation. Transformations from PIM to PSM are described in MDA so developers can obtain their low-level designs from high-level ones. Languages in MDA are called meta-models and are described using a common root meta-model: the Meta Object Facility (MOF) [omg, 2006]. The main advantage of MOF is that, once metamodels have been created, developers can benefit from model to model transformations (M2M) or model to text transformations (M2T) to obtain source code in an automatic way.

There are strong relations between Domain Specific Languages (DSLs) and models, specially as a generalization of MDE that may help to solve complex problems in more efficient ways [Kurtev et al., 2006]. DSLs are computer languages designed to fit a particular application domain. They are designed to be useful with a domain and the set of process it entails. Contrary to the more extended general purposed languages that are widely used across different domains, the syntax is usually small, although the division between them might be blurry sometimes. As DSLs have a clearly defined concrete problem domain it is able to represent the state of affairs in this domain. It introduces the basic abstractions of the domain and their mutual relations. When this abstract entity is explicitly represented as a model it becomes the reference model for the models expressed in the DSL, that is, it is a metamodel, the domain definition metamodel (DDMM). Which plays a key part in the definition of a DSL, letting us define a DSL as a set of coordinated models.

Another worth mentioning initiatives are the Eclipse Modeling Framework (EMF) [Steinberg et al., 2008] which is a basic MOF implementation; Xtext [Efftinge and Völter, 2006] which is a framework to create textual representations and notations from visual models and metamodels; and MOFScript [Oldevik, 2006] which provides a template language to perform M2T transformations. Xtext is a recent tool that facilitates the creation of new Domain Specific Languages (DSLs) and provides some interesting and useful characteristics such as code completion, syntax error checking and syntax highlighting among others. Moreover, its integration with EMF allows Xtext models to be represented as Graphical Modeling Framework [omg, 2006] visual models at any moment. The textual model representations from Xtext allows developers to build their DSLs as they usually do when working with any other programming languages, textually, and switch to visual models when necessary. d

When it comes to robotics a lot of effort has been placed to develop tools and provide bet-

ter software scalability and reusability. The modules developed through component oriented programming have a life cycle that can sometimes be of high complexity. This includes all common activities that are present during the life-cycle of any program: requirements analysis, design, implementation, unit testing, system integration, verification and validation, operation support, maintenance and disposal. The life-cycle of robotics modules is always in continuous development due to constant requirement changes [Brugali and Scandurra, 2009]. This makes the modification of structural properties of these modules a common thing during any time of their life-cycle [Brugali and Shakhimardanov, 2010]. In these cases, it generally entails performing changes in middleware-related code continuously. To help developers in these tasks, specific tools are required. Model Driven Engineering can provide robotics developers with the specific tools needed to automate the continuous changes in middleware-related code.

Because software development for robots can benefit from the use of MDA-Based tools, there are several robotics software examples making use of this techniques. An interesting example of this is the CoSMIC visual toolkit [Gokhale et al., 2002]. It is a complete open source MDA tool that allows the visual design, deployment and configuration of components based on the CORBA Component Model (CCM) [Wang et al., 2001]. However, CCM and its associated specifications are not widely used. In spite of that, the OMG aims at their future adoption within the robotics community by standardizing its novel Robotic Technology Component Specification (RTC) [OMG, 2008], which focuses on the structural and behavioral features required by robotics software as a supplement to a general component model. In this vein, it is worth noting OpenRTM-aist [Ando et al., 2011], a free RTC implementation that has appeared recently. OpenRTM-aist is a framework for robotics that provides developers with a set of tools to create and manage components. Two of these tools are Eclipse-based GUI tools: RTBuilder and RTSystemEditor. RTBuilder allows developers to create components defining their names, connectors (ports), parameters, programming language or their operating system. This tool can then be used to automatically generate source code templates. RTSystemEditor provides a mechanism to edit and configure components which are registered on a known name service. RTSystemEditor can start, stop or reset components, add and remove links and use introspection capabilities to monitor components at run time.

Another interesting tool is the 3-View Component MetaModel (V3CMM) [Alonso et al., 2010]. It is a platform-independent modeling language for component-based applications that makes use of MOF-based metamodels. V3CMM provides three views that are loosely coupled and allow users to design a complete system. With V3CMM it is possible to model the static structure of components (named as structural view), the behavior of these components (coordination view) and to model the functionality of these components, described as algorithms (algorithmic view). Two of these views are based on UML [Rumbaugh et al., 2004]:

1. The coordination view uses a simplified version based on UML state machines in order to model the different states of a component
2. The algorithmic view, based on UML activity diagrams, executes a specific behavior depending on the current state of the component

The structural view is used to define components and their dependencies by specifying both its required and provided interfaces. Once users model the system using these three views, it is possible to perform M2M to reduce the level of abstraction and M2T transformations to automatically obtain the final source code.

The SmartSoft [Schlegel et al., 2010] robotics framework also provides an MDA tool based on a UML profile implementation. The development process starts modeling an idea at a high-level of abstraction. This model is then refined through several transformations to obtain the software components source code. First, developers have to describe the system in a model independent platform (PIM) where information about middleware, operating system, programming languages and other properties are unknown. Then the PIM is transformed to a platform-specific model (PSM), where details about middleware or operating systems are specified. This PIM is also transformed to a platform-specific implementation (PSI) where developers can add their code and libraries. The next step is to deploy the components. In this vein SmartSoft uses the platform description model (PDM) to define the target platform properties. The model is extended with this platform information and finally the system can be run following the specified deployment model. During the development process users can guide the transformations in order to obtain a specific component by selecting the desired real-time and QoS properties of the component and the communication middleware it will use. A set of well-defined communication patterns provides the necessary abstraction from the final communication model and its reference implementation. Currently, it supports the ACE [Schmidt and Huston, 2002] (SmartSoft/ACE) and ACE/-TAO [Schmidt et al., 1997] (SmartSoft/CORBA) communication middlewares and provides other interesting features, such as a mechanism to guarantee real-time properties using an external scheduler analyzer or dynamic wiring for components.

A.4 Robotic Frameworks

In an effort to bring all these features to developers, software engineering techniques has been put together into robotic frameworks. There are different frameworks available for robotics development. They all try to provide a supporting layer for modules development and communications in order to ease the tedious tasks involved in the whole development cycle (programming, testing, bug fixing...). However each one of them offers different characteristics which lead to some advantages and disadvantages on each of them.

These are the most well known and used robotics frameworks:

- Artoo [art, 2017]: A Ruby microframework for robotics and physical computing with support for 15 platforms.
- CLARATy [Volpe et al., 2001]: A robotics software developed by the Jet Propulsion Laboratory from NASA as part of the Mars program.
- EEROS [Brown, 2015]: An Easy, Elegant, Reliable, Open and Safe Real-Time Robotics Software Framework for the development of educational and industrial robots.
- JAUS [Rowe and Wagner, 2008]: A framework by the United States Department of Defense to develop an open architecture for the domain of unmanned systems.
- JDE [Canas et al., 2013]: A robotics and computer vision framework developed in C++ and using ICE as a middlewae to provide support for a distributed component-based programming environment. It also includes several tools and libraries.
- leJOS NXJ [Bagnall, 2011]: An open-source Java programming environment for the Lego NXT robot kit.

- MOOS [Newman, 2008]: A C++ cross platform framework for robotics research.
- MyRobotLab [myr, 2017]: An open source Java service based framework for robotics control.
- OpenRTM-aist [Ando et al., 2008]: A robotics middleware developed by the Japanese National Institute of Advanced Industrial Science and Technology. It is based on the RT middleware standard [Ando et al., 2005].
- Orca [Makarenko et al., 2006]: An open source framework for developing component based robots.
- OROCOS [Bruyninckx et al., 2003]: C++ framework for component-based robot control software.
- RoboComp [Gutiérrez et al., 2013]: An open-source robotics framework providing the tools to easily create and modify software components that communicate through public interfaces. More details on this framework are given in section ??.
- Rock [Joyeux et al., 2014]: The Robot Construction Kit, a software integration framework for robotic systems based on Orocos/RTT. It was mainly designed for space robotics and it uses a model driven approach to handle the complexity of component networks which can reconfigure at run-time.
- ROS (Robot Operating System) [Quigley et al., 2009] provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management and more. Currently, ROS 2, a completely rebuild of this framework is under development.
- URBI [Baillie, 2005] A robotics framework based on the UObject distributed C++ component architecture. It uses the urbiscript orchestration language which is a parallel and event-driven script language.

Since some of these frameworks do not offer the whole set of tools needed for robotics development, some of them rely on other projects for that. Sometimes it has even been decided to keep separated projects in order to make them more independent and usable by different communities of developers. *e.g.*, the Gazebo project [Koenig and Howard, 2004] is an independent multi-robot simulator used as main simulator of the Robotics Operating System (ROS) [Quigley et al., 2009], which is even developed by the same team (the Open Source Robotics Foundation, OSRF). These are the main tools that compliment the main robotics framework and help in the development and testing of robotics:

- Player [Gerkey et al., 2003]: A robot control interface, often used along with its simulation backends, Stage and Gazebo.
- CARMEN [Montemerlo et al., 2002]: Carnegie Mellon Robot Navigation Toolkit, collection of modular software for mobile robot navigation.

- Gazebo [Koenig and Howard, 2004]: A robotics simulator that allows users to simulate populations of robots in complex indoor and outdoor environments. It contains physics engine, high-quality graphics and graphical interfaces. It is often the main simulator used in combination with the ROS framework.
- LSTS Toolchain [Pinto et al., 2013]: is a set of tools and frameworks for the development of Networked Vehicle Systems.
- miniBloq [da Silva Gillig, 2017]: A graphical programming interface to program robotic boards (Arduino Compatibles).
- Open Dynamics Engine [Smith et al., 2005]: A library for simulating rigid body dynamics. It has a C/C++ API and integrates collision detection with friction.
- OPRoS [Jang et al., 2010]: The Open Platform for Robotic Services is a component based open source platform for robotics with Integration Development Environment (IDE) tools, a server, and a test and verification tool.
- Robot Overlord [rob, 2017]: A multi-robot simulator written in Java and OpenGL.
- Simbad [Hugues and Bredeche, 2006]: Is a robot simulator written in JAVA.
- STDR Simulator [std, 2017]: The Simple Two Dimensional Robot Simulator is a multi-robot 2D simulator implemented through a distributed, server-client based architecture. It also provides a GUI developed in QT, for visualization purposes and some other functions.
- TeamBots [Balch, 2002]: A Java-based collection of application programs and Java packages for multiagent mobile robotics research.

A.4.1 RoboComp

RoboComp is a free software robotics framework (licensed under General Public License, GPL). It is a component oriented framework, that supports a wide range of different platforms and languages. It is model-driven and built around three key elements: a component model, a communications middleware and a set of tools that facilitates the writing and maintaining of robotics code. It started in 2005 as a way to create and reuse code written by many different people and that was meant to be used in many different robots. The central idea is to define a processing and coding entity that can be created and maintained largely decoupled from the rest of the system. These units or components are full fledged processes when running and occupy its own subdirectory in the global code repository. They communicate with other components using a public interface and through an underlying communications middleware. Building on this generic idea, RoboComp is now the result of many years of further elaboration and adaptation to our everyday research and engineering activity and, nevertheless, many more improvements are in the way now due to the increasing complexity of current robots and their control and cognitive architectures. The repository holds now more than one hundred components, along with classes and tools specifically designed to improve and ease the robotics software designer experience. It covers functionalities of different robotics and artificial vision topics mainly through integration of third party libraries.

RoboComp has its own component model, inspired by the ORCA model and making it evolve to fit robotics needs along years of development. As a middleware, RoboComp primarily uses ICE/ZeroC [ZeroC, 2016] and there is ongoing experimental work to make RoboComp middleware agnostic, so its components can be re-generated to use other middlewares. Actually currently has support to communicate with ROS components.

A.4.1.1 Component Model

The component model is fairly simple and easy to use. Components may implement, subscribe, or publish interfaces in order to communicate among them. Two main Domain Specific Languages (DSLs) that have been created to define a component at a very high level of abstraction. An "Interface Definition Specific Language" (IDSL) is used to describe the component Interfaces. With IDSL you write the data structures and functions that a component can implement, require, subscribe to or publish. A component can implement several interfaces, offering different views of its internal functioning. Also, the same interface can be implemented by many components. It currently corresponds to a subset of Ice's Slice interface language. CDSL stands for "Component Definition Specific Language" and allows the user to specify its name, accessible interfaces, communication connections, target language and other available modules or libraries that you want to include in the building scripts. The design and implementation of this DSLs was part of a collaborative work among developers furtherer explained in [Romero-Garces et al., 2011]. Using these two DSLs, RoboComp can generate the source code of the component using a tool designed to this end. The complete, functioning code of a component is created ready to be compiled and executed. We use a smart inheritance mechanism to separate the generic stuff from the user specific stuff and, based on it, the next time you generate a component, your code will remain untouched but access to new defined proxies will be there.

A.4.1.2 Libraries, Tools and Files

RoboComp also provides extra libraries and tools. Different libraries and classes have been developed to ease the components development. Among them, the proprioception library stands out, called InnerModel. It plays an important role in the development of robotics components as it helps with representation and geometric transformation between different reference frames. It is based on an XML description of the robot kinematic stored as a file. In the file all the joints and links of the kinematic tree should be described. The library also provides methods to estimate projections and frame transformations.

Several tools are provided along with the core of RoboComp. These tools are meant to help perform different operations mostly over components such as monitoring, record and replay or automated generation from the DSL specifications. Specially interesting and extensively used is the robocomp simulation tool (rcis) used to simulate robots and environments to make the initial testing of algorithms easy and safe. Figure A.2 shows a screenshot of the simulator with a model of the robot in an apartment with five tables and objects on them.

A set of binary files are provided through the git-annex tool. This tool is used to avoid the overhead that the store of binaries can cause in git repositories. Using this tool all the extra needed binary files can be stored to share them with the rest of developers. Among them you can find files such as datasets for testing, 3D models for simulation purposes or trained models

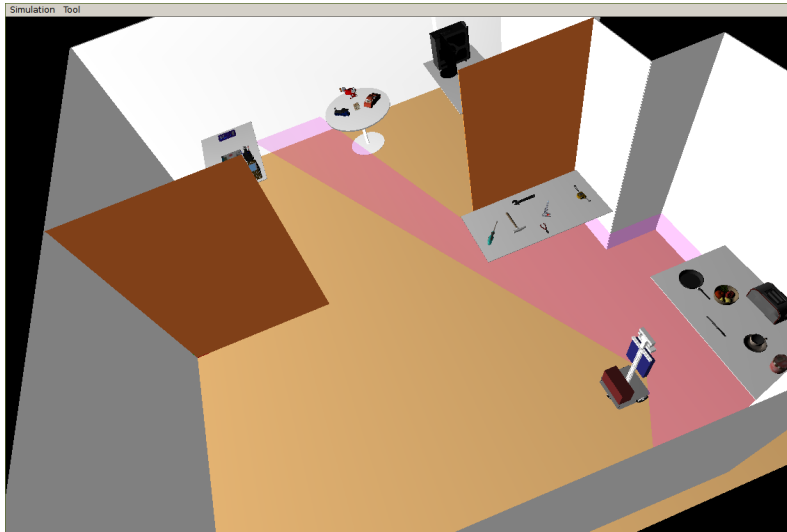


Figure A.2: RoboComp simulation tool with a model of the robot in an apartment with 5 tables and objects on them.

for Deep Neural networks.

As RoboComp was the robotic framework used for the support of this thesis. Therefore, mostly all the research and development showcased here is publicly available under RoboComp's repository¹. Publications such as [Romero-Garces et al., 2011], [Gutiérrez et al., 2013] or [Gutiérrez et al., 2013] are prove of these developments. Extra contributions not covered here have been done, specially notable are those made through the administration and mentoring of several editions of the Google Summer of Code programme.

A.5 Robot Perception Software

Part of the software development of this dissertation was made in the form of open source collaboration to perception software. Here we review the two main visual libraries that have been contributed, the Point Cloud Library [Rusu and Cousins, 2011] with a strong focus on 3D information processing and OpenDetection [Sarkar and Gutierrez, 2016], a library that combines the use of geometry and texture information in a unified API.

Robots working in unstructured environments have to perceive the world in order to act accordingly. The visual perception is still a field open for research. Lots of problems remain unsolved and to develop solutions for them the use of the latest development becomes a need. Therefore the need of reuse and easy and fast deployment of state-of-the-art solutions.

One of the main wanted characteristics in software engineering is software reusability. This is specially needed when it comes to research algorithms and specially for robotics where the complexity of problems makes it impossible to develop a whole system on your own .

Since the main focus of this dissertation is in perception, along with the specific research described here, some contributions to visual perception software have been made.

¹<https://github.com/robocomp>

A.5.1 Point Cloud Library

The Point Cloud Library was born right after the massive production of 3D sensors which lead to their price drop. The use of these devices became more and more common among roboticists and the research and development involving point clouds grew significantly.

PCL is built using modern C++ and written with a focus on high performance making the most use of modern CPUs and GPUs. It makes extensive use of templates and the underlying data structures use Streaming SIMD Extensions for optimization. For the mathematical operations the Eigen open source library for linear algebra is used [Guennebaud et al., 2010]. OpenMP (see <http://openmp.org>) and Intel Threading Building Blocks (TBB) [Reinders, 2007] library are supported for parallelization purposes. Boost libraries [Karlsson, 2005] are used for shared pointers management.

PCL contains numerous 3D processing algorithms that operate on point cloud data. The number of these algorithms are continuously growing as the library has numerous vision researches contributing and updating the algorithms. Among them you can find: filtering, feature estimation, surface reconstruction, model fitting, segmentation, registration and more. The algorithms are implemented through base classes in an effort to make them share common functionalities that keep the implementation of them clean and compact. This way common pipelines can be applied to different algorithms.

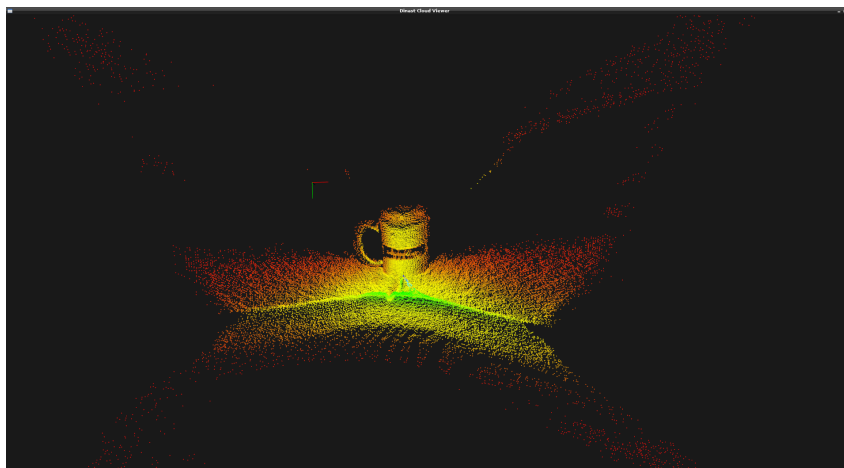


Figure A.3: Point Cloud from a cup on a table obtained using the PCL Dinast grabber.

As part of the development on paper 7, a new driver for Dinast Cameras was produced for PCL 1.7. This driver makes use of the generic grabber interface present in the library since PCL 1.0. It was tested with IPA-1110, Cyclopes II and the IPA-1002 ng T-Less NG cameras however it is expected to work accordingly with the rest of Dinast devices since development was supervised and approved by the manufacturer. Figure [?] shows the point cloud of a cup on a table obtained using the developed PCL grabber. This work was performed under the Dinast Code Sprint programme. For further information on how to make use of this driver please refer to the official tutorial in [Gutiérrez, 2015]

Part of work on chapter 6 was also a development for the PCL. The superquadrics fitting and point cloud completion research was produced under the 2014 Google Summer of Code programme. For more information on the progress of this work you can check the development blog in [Quispe and Gutiérrez, 2014].

A.5.2 OpenDetection

Open Detection (OD) [Sarkar and Gutierrez, 2016] is a standalone open source project for object detection and recognition in images and 3D point clouds. Open Detection is released under the terms of the BSD license. The project was originated as part Google Summer of Code 2015 programme with the aims of having a vision tool for robotics (in particular for the RoboComp framework).

The library is built with a very specific goal to answer the fundamental problem of Computer Vision Object Recognition and Detection. It is meant to make the existing algorithms to process images and 3D data available to everyone in a common, intuitive and user-friendly API.

The basic classes in OD are `Trainers` and `Detectors`. A `Trainer` (the offline stage) of a detection method acts on training input data to produce an intermediate output called trained data. A corresponding `Detector` (the online stage) of the same method uses the trained data produced by `Trainer` to detect or recognize objects in a given `Scene`. A `Scene` is a structure that contains data sensed from a specific view of the robot (*e.g.*, image or point cloud). Trained data is usually stored in a preconfigured directory structure depending on the method starting from the base directory set for OpenDetection.

The data produced by a `Trainer` can be used by any of the `Detector` classes. The `Detector` can use data of different types of `Trainers` (or no trainers at all). Therefore there is many-to-many mapping between trainers and detectors which is currently resolved by Documentation. The documentation provides the information regarding what `Trainer` to use for a given `Detector`.

Each `Trainer` implements a virtual function `train` with the following signature:

```
1 virtual int train() = 0;
```

Each `Detector` implements two functions of the following signature:

```
1 detect ()
2 detectOmni ()
```

`detectOmni()` performs a detection/recognition on the entire scene (unsegmented and unprocessed) and provides information about the detection as well as its exact location. `detect()` takes an 'object candidate' or a segmented/processed scene as an input and identifies if the entire scene is a detection.

```
1 virtual ODDetections* detect(ODScene *scene);
2 virtual ODDetections* detectOmni(ODScene *scene);
```

Depending on the type of provided by the scene, detectors are categorized in `od::Detector2D` and `od::Detector3D`.

The result of a `Detector` is returned as a `Detection`. A detection contains detection/recognition details as well as location information within the scene (for example bounding box for `od::ODDetection2D` and location/orientation for `od::Detection3D`)

An example of a typical code covering most of the pipeline looks like:

```
1 //train:
2 od::g2d::ODHOGTrainer *trainer = new od::g2d::ODHOGTrainer("", trained_data_dir);
3 trainer->setPosSamplesDir(pos_samples);
4
```

```
5 | //set all the configurations as required by the trainer, default values are also provided
6 | trainer->setNegSamplesDir(neg_samples);
7 | trainer->setNOFeaturesNeg(10);
8 | trainer->setTrainHardNegative(true);
9 | trainer->train(); //train!
10 |
11 | //detect:
12 | ODDetector *detector = new od::g2d::ODHOGDetector; //chose a detector type
13 | detector->setTrainingDataLocation(trained_data_dir);
14 | detector->init(); //init with the required options
15 |
16 | //do as may detections as needed in a loop using the initialized settings
17 | //Use the detect* methods for detection. scene is a Scene object from frameGenerator
18 | ODDetections2D *detections = detector->detectOmni(scene);
19 |
20 | //infer
21 | showimage(detections->renderMetaInfo(*scene).getCVImage()) //do something with detections,
```

This library is continuous development and new algorithms and improvements are added frequently. For further documentation you can refer to the examples and tutorials provided along with the source code of the library. Also a workshop video is available on the official website².

²<https://www.youtube.com/watch?v=d7JWTqsJ5DQ>

Appendix B

Publications not covered in this thesis

2016:

- “A Multimodal Control Architecture for Autonomous Unmanned Aerial Vehicles”. **Marco A. Gutiérrez**, Luis Fernando D’Haro, Rafael E. Banchs. HAI ’16 Proceedings of the Fourth International Conference on Human Agent Interaction, Pages 107-110, October, 2016.

2015:

- “Multi-robot collaborative platforms for humanitarian relief actions”. **Marco A Gutiérrez**, Suraj Nair, Rafael E Banchs, Luis Fernando D’Haro Enriquez, Andreea I Niculescu, Aravindkumar Vijayalingam. Humanitarian Technology Conference (R10-HTC), 2015 IEEE Region 10. pp. 1-6. 9-12 December 2015. Cebu, Philippines.
- “Enhancing Multimodal Embeddings with Word Semantic Relations for Image Search Applications”. **Marco A. Gutiérrez**. International Workshop on Embeddings and Semantics (IWES 2015). 15 September 2015. Alicante, Spain.
- “Gualzru’s path to the Advertisement World”. Fernando Fernández, Moisés Martínez, Ismael García-Varea, Jesús Martínez-Gómez, Jose Pérez-Lorenzo, Raquel Viciano, Pablo Bustos, Luis Manso, Luis Calderita, **Marco Gutiérrez**, Pedro Núñez, Antonio Bandera, Adrián Romero-Garcés, Juan Bandera, Rebeca Marfil. Proceedings FinE-R Workshop, IROS 2015. pp. 55-65. 2 October, 2015. Hamburg, Germany.

2014:

- “Ursus Team - Team Description Paper”. **Marco A. Gutiérrez**, LJ Manso, LV Calderita, P Bustos, F Cid, M Paoletti, A Sánchez, JP Bandera, J Martinez, M Martinez, J Garcia. RoCkiN Robotics Challenge, 26 - 30 November 2014, Toulouse, France.

2013:

- “Recent Advances in RoboComp”. **Marco A Gutiérrez**, A Romero-Garcés, P Bustos, J Martínez. Journal of Physical Agents. Special Issue on advances on Physical Agents. Vol. 7, No. 1 pp. 38-47, 2013.

2012:

- “The spatial change detection problem in robotics: a probabilistic approach based on mixture of Gaussians”. **Marco A. Gutiérrez**, L Manso, P Drews Jr, P Nunez. 5th International Conference on Spatial Cognition (ICSC2012), September 2012. Roma, Italia.
- “Progress in RoboComp.” **Marco A. Gutiérrez**, A. Romero-Garcés, P. Bustos, J. Martínez In Proc. of Workshop of Physical Agents, WAF 2012, September 2012. Santiago de Compostela, Spain.
- “Graph Grammars for Active Perception”. Lj Manso, Pablo Bustos, Pilar Bachiller, **Marco A. Gutierrez**. Proc. of 12th International Conference on Autonomous Robot Systems and Competitions ISSN 978-972-98603-4-8, pp 63-68. April 2012.

2011:

- “Improving the life cycle of robotics components using Domain Specific Languages”. A. Romero-Garces, L.J. Manso, **Marco A. Gutiérrez**, R. Cintas and P. Bustos.. In 2nd International Workshop on Domain-Specific Languages and models for ROBotic systems (DSLRob’2011). September 2011. San Francisco, USA.

Bibliography

- [OMG, 2008] (2008). Object management group, robot technology component specification. <http://www.omg.org/spec/RTC>, Last accessed on Dec 31 2016.
- [kin, 2016] (2016). Kinect oficial driver, microsoft corp. <https://www.microsoft.com/en-us/download/details.aspx>, Last accessed on Dec 30, 2016.
- [sch, 2016] (2016). Motors drivers, schunk gmbh & co. kg. https://br.schunk.com/br_en/services/tools-downloads/software/, Last accessed on Dec 30, 2016.
- [jet, 2016] (2016). Nvidia jetson tx1. <https://www.nvidia.com/object/jetson-tx1-module.html>, Last accessed on Dec 30, 2016.
- [art, 2017] (2017). artoo, ruby on robotics. <http://artoo.io>, Last accessed on Feb 19, 2017.
- [myr, 2017] (2017). Mrl, myrobotlab. <http://myrobotlab.org>, Last accessed on Feb 19, 2017.
- [rob, 2017] (2017). Robot overlord. url<https://github.com/MarginallyClever/Robot-Overlord-App>, Last accessed on Feb 19, 2017.
- [std, 2017] (2017). Simple two dimensional robot (stdr) simulator. url<http://stdr-simulator-ros-pkg.github.io/>, Last accessed on Feb 19, 2017.
- [Alonso et al., 2010] Alonso, D., Vicente-Chicote, C., Ortiz, F., Pastor, J., and Alvarez, B. (2010). V3cmm: A 3-view component meta-model for model-driven robotic software development. *Journal of Software Engineering for Robotics*, 1(1):3–17.
- [Ando et al., 2011] Ando, N., Kurihara, S., Biggs, G., Sakamoto, T., Nakamoto, H., and Kotoku, T. (2011). Software deployment infrastructure for component based rt-systems. *Journal of Robotics and Mechatronics*, 23(3):350–359.
- [Ando et al., 2005] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T., and Yoon, W.-K. (2005). Rt-middleware: distributed component middleware for rt (robot technology). In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3933–3938. IEEE.
- [Ando et al., 2008] Ando, N., Suehiro, T., and Kotoku, T. (2008). A software platform for component based rt-system development: Openrtm-aist. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 87–98. Springer.

- [Ayache and Lustman, 1991] Ayache, N. and Lustman, F. (1991). Trinocular stereovision for robotics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1).
- [Bagnall, 2011] Bagnall, B. (2011). *Intelligence unleashed: Creating LEGO NXT robots with Java*. Variant Press.
- [Baillie, 2005] Baillie, J.-C. (2005). Urbi: Towards a universal robotic low-level programming language. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 820–825. IEEE.
- [Balch, 2002] Balch, T. (2002). The teambots environment for multi-robot systems development. *Working notes of Tutorial on Mobile Robot Programming Paradigms, ICRA*.
- [Bird et al., 2009] Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- [Bohg et al., 2011] Bohg, J., Johnson-Roberson, M., León, B., Felip, J., Gratal, X., Bergstrom, N., Kragic, D., and Morales, A. (2011). Mind the gap: Robotic grasping under incomplete observation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*.
- [Bradski, 2000] Bradski, G. (2000). Open source computer vision library (opencv). *Dr. Dobb’s Journal of Software Tools*.
- [Brown, 2004] Brown, A. W. (2004). Model driven architecture: Principles and practice. *Software and Systems Modeling*, 3(4):314–327.
- [Brown, 2015] Brown, N. (2015). *Expanding the Impact of the EEROS Open Source Robotics Framework*. PhD thesis, Worcester Polytechnic Institute.
- [Brugali and Scandurra, 2009] Brugali, D. and Scandurra, P. (2009). Component-based robotic engineering. part i: Reusable building blocks. *IEEE Robotics and Automation Magazine*, 16(4):84–96.
- [Brugali and Shakhimardanov, 2010] Brugali, D. and Shakhimardanov, A. (2010). Component-based robotic engineering. part ii: Models and systems. *IEEE Robotics and Automation Magazine*, 17:100–112.
- [Bruyninckx et al., 2003] Bruyninckx, H., Soetens, P., and Koninckx, B. (2003). The real-time motion control core of the orocos project. In *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*, volume 2, pages 2766–2771. IEEE.
- [Bustos et al., 2016] Bustos, P., Manso, L., Bandera, J., Romero-Garcés, A., Calderita, L., Marfil, R., and Bandera, A. (2016). *A unified internal representation of the outer world for social robotics*, volume 418.
- [Canas et al., 2013] Canas, J., González, M., Hernández, A., and Rivas, F. (2013). Recent advances in the jderobot framework for robot programming. In *Proceedings of RoboCity2030 12th Workshop, Robótica Cognitiva*, pages 1–21.
- [Chen and Medioni, 1992] Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155.

- [da Silva Gillig, 2017] da Silva Gillig, J. (2017). Minibloq. <http://minibloq.org>, Last accessed on Feb 19, 2017.
- [Drews et al., 2010] Drews, P., Núñez, P., Rocha, R., Campos, M., and Dias, J. (2010). Novelty detection and 3d shape retrieval using superquadrics and multi-scale sampling for autonomous mobile robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3635–3640. IEEE.
- [Duncan et al., 2013] Duncan, K., Sarkar, S., Alqasemi, R., and Dubey, R. (2013). Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4238–4243. IEEE.
- [Efftinge and Völter, 2006] Efftinge, S. and Völter, M. (2006). oaw xtext: A framework for textual dsls. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32, page 118.
- [Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.
- [Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.
- [Gerkey et al., 2003] Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323.
- [Girshick et al., 2016] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158.
- [Gokhale et al., 2002] Gokhale, D. C. S. A., Natarajan, R., Neema, E., Bapty, T., Parsons, J., Gray, J., Nechypurenko, A., and Wang, N. (2002). Cosmic: An mda generative tool for distributed real-time and embedded component middleware and applications. In *In Proceedings of the OOPSLA 2002 Workshop on Generative Techniques in the Context of Model Driven Architecture*. ACM.
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [Guennebaud et al., 2010] Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. <http://eigen.tuxfamily.org>.
- [Gutiérrez, 2015] Gutiérrez, M. A. (2015). The pcl dinast grabber framework. http://pointclouds.org/documentation/tutorials/dinast_grabber.php.
- [Gutierrez et al., 2011] Gutierrez, M. A., Martinena, E., Sánchez, A., Rodríguez, R. G., and Nunez, P. (2011). A cost-efficient 3d sensing system for autonomous mobile robots. In *Proc. of XII Workshop of Physical Agents*.

- [Gutiérrez et al., 2013] Gutiérrez, M. A., Romero-Garcés, A., Bustos, P., and Martínez, J. (2013). Recent advances in robocomp. *Proceedings of the Workshop of Physical Agents (WAF2013)*.
- [Gutiérrez et al., 2013] Gutiérrez, M. A., Romero-Garcés, A., Bustos, P., and Martínez, J. (2013). Progress in RoboComp. *Journal of Physical Agents*, 7(1):38–47.
- [He et al., 2005] He, J., Li, X., and Liu, Z. (2005). *Component-based Software Engineering: the Need to Link Methods and their Theories*, pages 70–95. Proc. of ICTAC 2005, Lecture Notes in Computer Science 3722. Springer.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [Hugues and Bredeche, 2006] Hugues, L. and Bredeche, N. (2006). Simbad: an autonomous robot simulation package for education and research. In *International Conference on Simulation of Adaptive Behavior*, pages 831–842. Springer.
- [Jang et al., 2010] Jang, C., Lee, S.-I., Jung, S.-W., Song, B., Kim, R., Kim, S., and Lee, C.-H. (2010). Opros: A new component-based robot software platform. *ETRI journal*, 32(5):646–656.
- [Joo et al., 2010] Joo, K., Lee, T.-K., Baek, S., and Oh, S.-Y. (2010). Generating topological map from occupancy grid-map using virtual door detection. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–6. IEEE.
- [Joyeux et al., 2014] Joyeux, S., Schwendner, J., Roehr, T. M., and Center, R. I. (2014). Modular software for an autonomous space rover. In *The 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014)*.
- [Karlsson, 2005] Karlsson, B. (2005). *Beyond the C++ standard library: an introduction to boost*. Pearson Education.
- [Kazhdan et al., 2006] Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Proc. of the Fourth Eurographics Symposium on Geometry Processing*.
- [Kiros et al., 2014] Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.
- [Koenig and Howard, 2004] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE.

- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [Kurtev et al., 2006] Kurtev, I., Bézivin, J., Jouault, F., and Valduriez, P. (2006). Model-based dsl frameworks. In *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications, OOPSLA '06*, pages 602–616, New York, NY, USA. ACM.
- [Makarenko et al., 2006] Makarenko, A., Brooks, A., and Kaupp, T. (2006). Orca: Components for robotics. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 163–168.
- [Manso et al., 2010] Manso, L., Bachiller, P., Bustos, P., Núñez, P., Cintas, R., and Calderita, L. (2010). *RoboComp: a Tool-based Robotics Framework*, pages 251–262. Springer.
- [Manso et al., 2015] Manso, L., Bustos, P., Bachiller, P., and Núñez, P. (2015). A perception-aware architecture for autonomous robots. *International Journal of Advanced Robotic Systems*, 12(174):13.
- [Manso et al., 2016] Manso, L., Calderita, L., Bustos, P., and Bandera, A. (2016). Use and advances in the active grammar-based modeling architecture. *Proceedings of the International Workshop on Physical Agents 2016*, pages 31–36.
- [Marfil et al., 2014] Marfil, R., Calderita, L. V., Bandera, J. P., Manso, L. J., and Bandera, A. (2014). Toward Social Cognition in Robotics : Extracting and Internalizing Meaning from Perception. In *Workshop of Physical Agents WAF2014*, number June, pages 1–12, Leon, Spain.
- [McIlroy, 1969] McIlroy, M. D. (1969). Mass produced software components. *NATO, Scientific Affairs Division*, pages 79–85.
- [McIlroy et al., 1968] McIlroy, M. D., Buxton, J., Naur, P., and Randell, B. (1968). Mass-produced software components. In *Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany*, pages 88–98. sn.
- [Meguro et al., 2007] Meguro, J.-i., Takiguchi, J.-i., Amano, Y., and Hashizume, T. (2007). 3d reconstruction using multibaseline omnidirectional motion stereo based on gps/dead-reckoning compound navigation system. *The International Journal of Robotics Research*, 26(6):625–636.
- [Mhalla et al., 2016] Mhalla, A., chateau, T., Gazzah, S., and Ben Amara, N. E. (2016). A faster r-cnn multi-object detector on a nvidia jetson tx1 embedded system: Demo. In *Proceedings of the 10th International Conference on Distributed Smart Camera, ICDSC '16*, pages 208–209, New York, NY, USA. ACM.
- [Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

- [Montemerlo et al., 2002] Montemerlo, M., Roy, N., and Thrun, S. (2002). Carnegie mellon robot navigation toolkit. *Software package, download at www.cs.cmu.edu/carmen*.
- [Moravec, 1996] Moravec, H. (1996). Robot spatial perception by stereoscopic vision and 3d evidence grids. *Perception*.
- [Newman, 2008] Newman, P. M. (2008). Moos-mission orientated operating suite. *Massachusetts Institute of Technology, Tech. Rep, 2299(08)*.
- [Oldevik, 2006] Oldevik, J. (2006). Mofscript eclipse plug-in: metamodel-based code generation. In *Eclipse Technology Workshop (EtX) at ECOOP*, volume 2006.
- [omg, 2006] omg (2006). *Meta Object Facility (MOF) Core Specification Version 2.0*.
- [Pinto et al., 2013] Pinto, J., Dias, P. S., Martins, R., Fortuna, J., Marques, E., and Sousa, J. (2013). The lsts toolchain for networked vehicle systems. In *OCEANS-Bergen, 2013 MT-S/IEEE*, pages 1–9. IEEE.
- [Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.
- [Quispe and Gutiérrez, 2014] Quispe, A. H. and Gutiérrez, M. A. (2014). Fitting segmented pointclouds to superquadrics online. <http://www.pointclouds.org/blog/gsoc14/ahuaman/index.php>.
- [Rangel et al., 2016] Rangel, J. C., Cazorla, M., García-Varea, I., Martínez-Gómez, J., Fromont, É., and Sebban, M. (2016). Scene classification based on semantic labeling. *Advanced Robotics*, 30(11-12):758–769.
- [Reinders, 2007] Reinders, J. (2007). *Intel threading building blocks: outfitting C++ for multi-core processor parallelism*. ” O’Reilly Media, Inc.”.
- [Romero-Garces et al., 2011] Romero-Garces, A., Manso, L. J., Gutiérrez, M. A., Cintas, R., and Bustos, P. (2011). Improving the life cycle of robotics components using Domain Specific Languages. In *2nd International Workshop on Domain-Specific Languages and models for ROBotic systems (DSLRob’2011)*.
- [Romero-González et al., 2017] Romero-González, C., Martínez-Gómez, J., García-Varea, I., and Rodríguez-Ruiz, L. (2017). On robot indoor scene classification based on descriptor quality and efficiency. *Expert Systems with Applications*, 79:181–193.
- [Rosenfeld, 1969] Rosenfeld, A. (1969). Picture processing by computer. *ACM Computing Surveys (CSUR)*, 1(3):147–176.
- [Rowe and Wagner, 2008] Rowe, S. and Wagner, C. R. (2008). An introduction to the joint architecture for unmanned systems (jaus). *Ann Arbor*, 1001:48108.
- [Rubner et al., 1998] Rubner, Y., Tomasi, C., and Guibas, L. J. (1998). A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE.

- [Rumbaugh et al., 2004] Rumbaugh, J., Jacobson, I., and Booch, G. (2004). *Unified Modeling Language Reference Manual, The*. Pearson Higher Education.
- [Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.
- [Rusu et al., 2010] Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J. (2010). Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE.
- [Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.
- [Sarkar and Gutierrez, 2016] Sarkar, K. and Gutierrez, M. A. (2016). Opendetection library (od). <http://opendetection.com>, Last accessed on Dec 30, 2016.
- [Scharstein et al., 2001] Scharstein, D., Szeliski, R., and Zabih, R. (2001). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision, 2001.(SMBV 2001). Proceedings. IEEE Workshop on*, pages 131–140. IEEE.
- [Schlegel, 2006] Schlegel, C. (2006). Communication Patterns as Key Towards Component-Based Robotics. *International Journal of Advanced Robotic Systems*, 3(1):49–54.
- [Schlegel et al., 2010] Schlegel, C., Steck, A., Brugali, D., and Knoll, A. (2010). Design abstraction and processes in robotics: from code-driven to model-driven engineering. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 324–335. Springer.
- [Schmidt and Huston, 2002] Schmidt, D. and Huston, S. D. (2002). *C++ Network Programming, Volume 2: Systematic Reuse with ACE and Frameworks*. Addison-Wesley Professional.
- [Schmidt et al., 1997] Schmidt, D. C., Gokhale, A., Harrison, T. H., Levine, D., and Cleeland, C. (1997). Tao: A high performance endsystem architecture for real-time corba. *IEEE Communications Magazine*, 14(2).
- [Shepherd and Buchstab, 2014] Shepherd, S. and Buchstab, A. (2014). Kuka robots on-site. In *Robotic Fabrication in Architecture, Art and Design 2014*, pages 373–380. Springer.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.
- [Smith et al., 2005] Smith, R. et al. (2005). Open dynamics engine.
- [Steinberg et al., 2008] Steinberg, D., Budinsky, F., Merks, E., and Paternostro, M. (2008). *EMF: eclipse modeling framework*. Pearson Education.

- [Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.
- [Temerinac et al., 2007] Temerinac, M., Reisert, M., and Burkhardt, H. (2007). Invariant features for searching in protein fold databases. *International Journal of Computer Mathematics*, 84(5):635–651.
- [Torabi and Gupta, 2010] Torabi, L. and Gupta, K. (2010). Integrated view and path planning for an autonomous six-dof eye-in-hand object modeling system. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4516–4521. IEEE.
- [Um et al., 2011] Um, D., Ryu, D., and Kal, M. (2011). Multiple intensity differentiation for 3-d surface reconstruction with mono-vision infrared proximity array sensor. *IEEE Sensors Journal*, 11(12):3352–3358.
- [Um et al., 2013] Um, D., Ryu, D., and Kang, S. (2013). A framework for unknown environment manipulator motion planning via model based realtime rehearsal. *Intelligent Autonomous Systems 12*, pages 623–631.
- [Vaughan et al., 2003] Vaughan, R. T., Gerkey, B. P., and Howard, A. (2003). On device abstractions for portable, reusable robot code. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, pages 2421–2427 vol.3.
- [Volpe et al., 2001] Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R., and Das, H. (2001). The clarity architecture for robotic autonomy. In *Aerospace Conference, 2001, IEEE Proceedings.*, volume 1, pages 1–121. IEEE.
- [Wang et al., 2001] Wang, N., Schmidt, D. C., and O’Ryan, C. (2001). Overview of the corba component model. In *Component-Based Software Engineering*, pages 557–571. Addison-Wesley Longman Publishing Co., Inc.
- [Woodman and Chun, 2006] Woodman, G. F. and Chun, M. M. (2006). The role of working memory and long-term memory in visual search. *Visual Cognition*, 14(4-8):808–830.
- [Yamauchi, 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA’97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151.
- [Yuret and De La Maza, 1993] Yuret, D. and De La Maza, M. (1993). Dynamic hill climbing: Overcoming the limitations of optimization techniques. In *The Second Turkish Symposium on Artificial Intelligence and Neural Networks*, pages 208–212. Citeseer.
- [ZeroC, 2016] ZeroC (2016). Internet communications engine (ice). <https://zeroc.com/products/ice>, Last accessed on May 10, 2017.