SAPIENZA
UNIVERSITÀ DI ROMA

Interactive Generation and Learning of Semantic-Driven Robot Behaviors

Roberto Capobianco

ID number 1509464

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Engineering in Computer Science

XXIX Cycle

Department of Computer, Control, and Management Engineering
Sapienza University of Rome
Rome, Italy

December 2016

Thesis Advisor

Prof. Daniele Nardi

Co-Advisor

Prof. Barbara Caputo

Review Committee

Prof. Nick Hawes
University of Birmingham

Prof. George Konidaris
Brown University

Thesis defended on February 21st, 2017
in front of a Board of Examiners composed by:

Prof. Alfonso Gerevini, *University of Brescia* (chairman)

Prof. Laura Tarantino, *University of L'Aquila*

Prof. Miriam Di Ianni, *University of Rome Tor Vergata*

Interactive Generation and Learning of Semantic-Driven Robot Behaviors

Keywords: Robot Learning, Semantic Mapping, Spatial Affordances, Reinforcement Learning, Policy Improvement, Dynamics Models, Knowledge Representation

Ph.D. thesis. Sapienza University of Rome

Version: February 15th, 2017

Website: http://robertocapobianco.com

Author's email: capobianco@dis.uniroma1.it

For my family:
Orietta, Simone,
Pia, Egidio and Chiara.

Abstract

The generation of adaptive and reflexive behavior is a challenging task in artificial intelligence and robotics. In this thesis, we develop a framework for knowledge representation, acquisition, and behavior generation that explicitly incorporates semantics, adaptive reasoning and knowledge revision. By using our model, semantic information can be exploited by traditional planning and decision making frameworks to generate empirically effective and adaptive robot behaviors, as well as to enable complex but natural human-robot interactions.

In our work, we introduce a model of semantic mapping [102], we connect it with the notion of affordances [46], and we use those concepts to develop semantic-driven algorithms for knowledge acquisition, update, learning and robot behavior generation. In particular, we apply such models within existing planning and decision making frameworks to achieve semantic-driven and adaptive robot behaviors in a generic environment. On the one hand, this work generalizes existing semantic mapping models and extends them to include the notion of affordances. On the other hand, this work integrates semantic information within well-defined long-term planning and situated action frameworks to effectively generate adaptive robot behaviors. We validate our approach by evaluating it on a number of problems and robot tasks. In particular, we consider service robots deployed in interactive and social domains, such as offices and domestic environments. To this end, we also develop prototype applications that are useful for evaluation purposes.

Acknowledgments

Throughout my Ph.D. studies, I had the privilege to work with extraordinary people that made this possible by encouraging me day after day.

First and foremost, I am thankful to my advisor, Daniele Nardi, for supporting my research and guiding my studies towards exciting problems and applications, as well as for aiding me to stay on the right path, while supplying enough freedom to explore my own way during this journey. I also thank Drew Bagnell, for welcoming me as a visitor in his lab (the LairLab) at the Carnegie Mellon University. It has been a great experience, that inspired me and provided countless insights and thoughts, introducing me with passion and excitement to new research fields and problems. In the same way, I am grateful to Geoff Gordon for allowing me to take part to his reading groups in the SELECT lab.

I would also like to thank my co-advisor, Barbara Caputo, for her comments and discussion especially at the early stages of this research , as well as George Konidaris and Nick Hawes for serving as reviewers on my thesis committee.

I am grateful for Giorgio Grisetti, for supporting me as one of his students, as well as for providing lots of hints and suggestions both during my coding and research activities. His recommendations have been essential for pursuing a Ph.D. and for better understanding computer vision applications. Similarly, I am thankful to Luca Iocchi, for his insights and for the helpful discussions about semantic maps (Chapters 4 and 6).

During this path, I had the chance to collaborate with great and friendly colleagues, whose contribution and help to this thesis is valuable on multiple aspects. I thank Jacopo Serafin, for contributing to some ideas (Chapter 4), sharing this journey and always working side by side with me. Also, I am very grateful to Francesco Riccio, for our work in the last year (Chapters 5 and 7), our ambitious goals, our ideas and the long hours of day and night work. I am sure that our collaboration will continue in the upcoming years. Thanks to Guglielmo Gemignani e Fabio Previtali for our long discussions, our collaboration at the beginning of my research and for the projects that we realized together. Additionally, I would like to thank Wen Sun and Arun Venkatraman for working together on dynamics learning applications during my 8 months at Carnegie Mellon (Chapter 8).

Being a member of the lab Ro.Co.Co. and the LairLab has been a wonderful experience: thanks to their members and the staff. Special thanks go to Emanuele Bastianelli, Daniele Bloisi, Shreyansh Daftry, Marco Imperoli, Andrea Pennisi, Andrea Vanzo and Jiaji Zhou. Also, I thank my office-mates, who shared the “Ph.D. room” with me for several years.

Last but not least, I am very grateful to my family. I am grateful to my mother, Orietta, for always supporting and doing everything for me, just by herself. You are my hero. I am thankful to my brother, Simone, for always giving me the energy for living, for smiling and for seeing the beautiful parts of each day. I thank my grandmother, Pia, and my departed grandfather, Egidio, for their help, their support, their faith in me and their never ending love. You are an example of life, my perfect model of the world. Finally, thanks to Chiara, for being the good person that she is, for being an extension of my family and for understanding me like no one else. Without even considering the past, your smile, your support and your presence are one of the most precious gifts of my life.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

We must think things not words, or at least we must constantly translate our words into the facts for which they stand, if we are to keep to the real and the true.

— Oliver Wendell Holmes Jr.

Robotic platforms and their applications are rapidly becoming consumer products, aimed at helping and supporting everyday life. The recent development of autonomous devices typically recalls the concept of *intelligent* agents, capable of smoothly interacting with humans, understanding their requests, and taking into account their needs, habits and interaction modes. To define and describe intelligence, meaningful traits such as the "adjustment or adaptation" to the environment [38], as well as "the ability to reason, plan, solve problems, think abstractly, comprehend complex ideas, learn quickly and learn from experience" [48] have been repeatedly identified by researchers and psychologists. Yet, generating such behaviors on robots and reproducing these qualities through static software[1] is generally difficult, mostly unsatisfactory and error prone in uncontrolled scenarios.

Many attempts to incorporate characteristics of reasoning, planning and adaptation in robots have been made, through automated decision making solutions (e.g., Markov decision processes [110]) or, more generally, by adopting *prescriptive* models that provide optimal decisions. However, these approaches typically rely on a plethora of parameters (e.g., the state of the environment) that must be modeled, but cannot be exhaustively uncovered, tracked or carved by robotic experts even through a continuous guess-and-check process [114, 120]. Additionally, such methods generally define a reactive paradigm of *situated action and learning*, where action and perception are continuously interleaved to achieve a goal that is intrinsically designed within the agent [135] (e.g., through a reward or cost function). Although this enables efficient control and decision making frameworks, situated approaches do not easily scale to dynamic uncontrolled scenarios, and hardly achieve both long timescale goal *deliberation* and complex *reflection* (i.e., self-adaptation, meta-reasoning) capabilities. Hence, their use comes at the cost of poor generalization

[1] By static software, we mean hand-written and pre-programmed software generated by an expert.

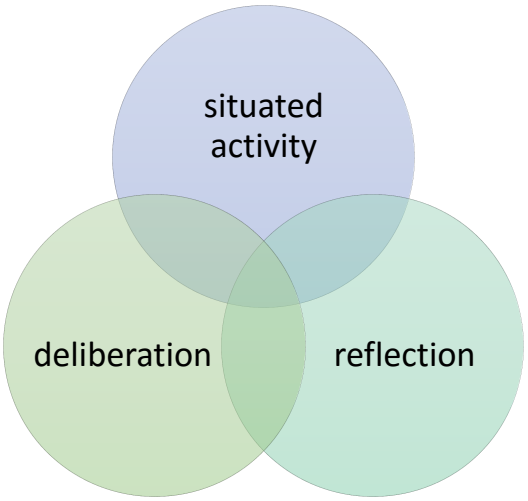


Figure 1.1. Interrelations between the problems of deliberation, situated action and reflection.

skills to novel settings and unintuitive behavior models, that are mostly disconnected from the semantics of the world and from common human-robot interaction patterns.

To tackle this problem, both explicit models as *semantic maps* [109, 115] and implicit representations, such as deep neural networks [79, 80, 88] have been recently adopted in robotics to capture world concepts and to model semantics. Despite enabling simple forms of reflection, semantic maps typically represent abstract knowledge and have little connection to decision making frameworks and their reactive paradigm. However, they explicitly capture (and expose) a semantic model of the world. For this reason, they are mostly adopted for high-level task planning and have almost no influence on situated action and decision making. Deep neural networks, instead, have been successfully adopted to implicitly model and encode world semantics, while directly determining agent policies [89, 143, 88]. Unfortunately, captured models are generally difficult to interpret, cannot be used within traditional planning frameworks and require large amounts of training data. Hence, the use of deep learning in robotics comes at the cost of giving away long timescale planning and data efficiency, in place of multiple and long data acquisition sessions that are potentially unfeasible for the robot (or must be simulated).

In our work, we develop a framework for knowledge representation, acquisition and behavior generation for robots that explicitly incorporates semantics, adaptive reasoning and knowledge revision, both for high-level planning and situated action in interactive environments. In this sense, our attempt consists in interconnecting the previously introduced problems of deliberation, situated action and reflection in an autonomous agent, as shown in Figure 1.1. Under this perspective, the central thesis of our research is the following:

> **Spatial semantics is necessary to obtain accurate and intuitive world models that enable both deliberation and situated activity. This type of information can be exploited by traditional planning and decision making frameworks to generate empiri-**

cally effective and adaptive robot behaviors, as well as to enable complex but natural human-robot interactions.

To validate this thesis, we introduce a model of semantic mapping, connect it with the notion of spatio-temporal affordance maps, use those concepts to develop semantic-driven algorithms for knowledge acquisition, update, learning and robot behavior generation, and apply such models and algorithms to prototype robotics systems deployed in highly interactive and social environments. More specifically, we introduce a **model for semantic mapping**, that extends and formalizes previous definitions [102], as well as the notion of **spatio-temporal affordance maps**. We apply such models within existing planning and decision making frameworks to achieve semantic-driven and adaptive robot behaviors in a generic environment. To obtain human-level interactions and to reduce the number of tracked parameters, we additionally adopt, in our robotic applications, intuitive and natural interaction modes. On the one hand, this work generalizes existing semantic mapping models and extends them to the the notion of affordance, typical of ecological robotics [32, 58]. On the other hand, this work integrates semantic information within well-defined long-term planning and situated action frameworks to effectively generate adaptive robot behaviors.

The remainder of this introduction is organized as follows. While the contributions of our work are explicated in Section 1.5, we first introduce a motivating example (Section 1.1) and we discuss the notions of deliberation, situated action (Section 1.2), prescription and reflection (Section 1.3). Then, we describe the general approach of this thesis (Section 1.4) and the prototype systems on which we evaluate our ideas (Section 1.6). Finally, we describe the structure of this thesis in Section 1.7.

1.1. Motivating Example

To motivate our work, we present a robotic example, that illustrates the fundamental issues that justify the use of semantic information at multiple scales, both for deliberation and situated activity, as previously described.

For the purposes of our example, we introduce Sandy. Sandy is a wheeled holonomic mobile robot which can manipulate objects through a robotic arm mounted on its base, and can perceive the surrounding environment as well as recognize objects through a camera mounted on its arm. Sandy lives in a simple world, illustrated in Figure 1.2, that is composed of three platforms (A, B, and C), where it verbally and physically interacts with a human, Daniele. Typically, Daniele uses platform A for placing his slotted screwdrivers, platform B for keeping cross-head screwdrivers and platform C for storing hammers. However, Daniele is always thinking about new scientific discoveries, and sometimes he leaves his slotted screwdrivers on platform B.

When Daniele asks the robot to *"bring the slotted screwdriver on the platform"*, the robot faces a very ambiguous command [10]. To properly execute this command, Sandy needs to understand the semantics of its environment and explicitly reason about Daniele's preferences for storing his tools. Additionally, Sandy must evaluate the actions that are afforded by the current situation. In particular, it needs to understand whether Daniele already has the screwdriver, and he wants the robot to

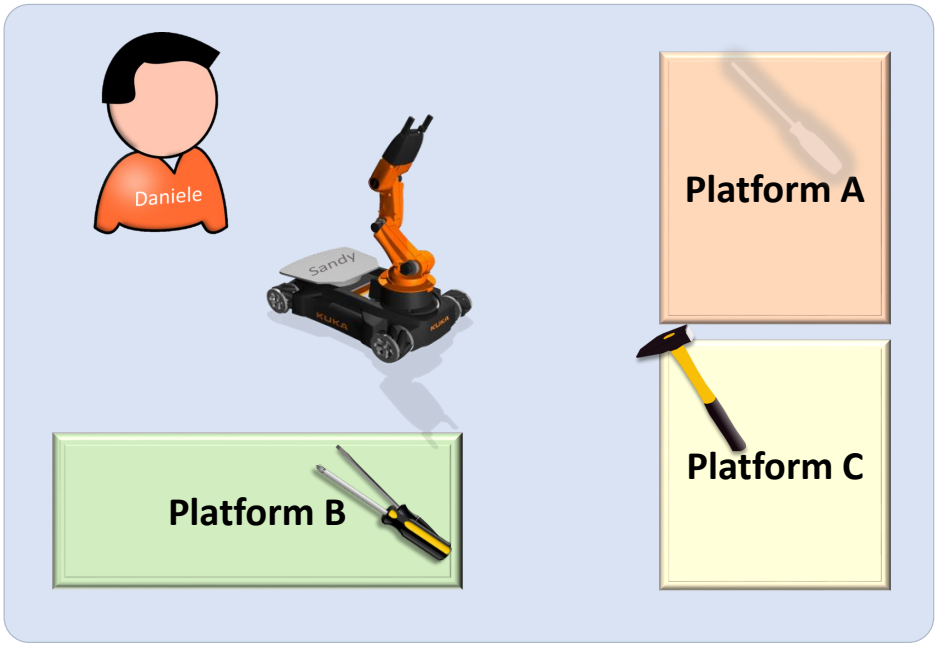


Figure 1.2. Motivating example: sketch of Sandy's environment.

bring it on a platform, or if he is asking for the tool, which is still on a platform. In the former case, Sandy can take the screwdriver, navigate to platform A and leave the object. In the latter case, the robot must navigate to platform A, presumably take the screwdriver and bring it to Daniele.

However, things are not as easy as they appear. Although Sandy does not know it yet, Daniele is often inaccurate, as he is focusing on his latest scientific discoveries. Hence, during the execution of its long-term plan, the robot eventually faces the situation in which the slotted screwdriver is not on platform A. At that moment, the generated plan is no longer supported by the current scenario and needs to be interrupted or modified. Still, Sandy can reason using its semantic knowledge and understand that B, which is also used for storing some screwdrivers, can occasionally hold the desired tool. Hence, the robot updates its knowledge about Daniele (and his distraction), modifies its plan to visit B and reconsiders the decision it needs to take. After finding the slotted screwdriver on B, Sandy can reach the position in the space that best allows (or affords) the object pick-up, and finally take the desired tool. To bring it to Daniele, the robot has to approach his human companion in a friendly way, and show that the object handover is finally possible. Intuitively, Sandy needs to evaluate Daniele's activities during time, his position in the environment and approach him from the front side. In other words, the robot needs to decide his motion in the best possible way (i.e., from the front) to support his planned action (i.e., perform the handover). Nevertheless, to start the handover, Sandy also needs to wait for the attention of Daniele. Only at that point the action is possible and the task is over.

Clearly, to successfully complete is assignment, Sandy increasingly exploited its environmental semantic knowledge, as well as its strongly adaptive reasoning

both for high-level planning and situated action. Hence, the goal of this thesis consists in contributing towards the development of a real world agent, endowed with capabilities that are similar to those of Sandy.

1.2. Deliberation and Situated Activity

Our claim is that spatial semantics is necessary to obtain accurate and intuitive world models that enable both deliberation and situated activity. Importantly, differentiating between deliberative and situated approaches, as well as integrating them in a robotic system, is essential to generate effective agent's behaviors. For this reason, we now review these two models and use them to motivate the perspective of this thesis.

Deliberation refers to the purposeful development of long-term courses of action that is carried out, typically through the use of a (symbolic) model of the world, in order to achieve some specified objectives. Often, a deliberative agent is practically realized through a top-down, four-step sense-model-plan-act [100, 42] architecture and is capable of dealing with complex situations. Under this perspective, in fact, the planning activity is considered as a form of problem solving, where tasks are decomposed into sub-tasks and a path is found from an initial state to a desired goal. Here, each path is an exact and clear specification of the agent's courses of action. Still, the applicability of this schema in robotics scenarios is strongly limited by the computational resources demanded for modeling and planning [60], by the uncertainty of the world against exact behavior specifications, as well as by the large amount of prior knowledge that the agent requires.

In contrast to deliberative theories, Suchman [137] introduces the concept of **situated action**, where plans do not model exact sequences specifying behavioral details, but rather represent action maps (i.e., a simplifications or sketches) that need concrete in-situ instantiatiations to agents' contingencies. In this sense, the planning problem is subsumed by the larger problem of situated action, that is characterized as follows:

> *That term underscores the fact that the course of action depends in essential ways upon the action's circumstances. Rather than attempting to abstract action from its circumstances and reconstruct it as a rational plan, the approach is to study how people use their circumstances to achieve intelligent action. Rather than build a theory of action out of a theory of plans, the aim is to investigate how people produce and find evidence for plans in the course of situated action. (Suchman, 1987)*

In robotics, such paradigm is typically realized through reactive behavior-based control architectures [14, 2, 60] and hybrid control architectures [3, 27, 111, 26]. The former are implemented in a bottom-up fashion to immediately consider sensory information. Reactive architectures, however, often fail at integrating world knowledge and generalizing to complex situations. The latter are based on layered partitions of deliberative and reactive functionalities. Hybrid approaches attempt to combine the benefits of deliberation and reaction, while reducing their specific inconveniences. This is obtained, for example, by binding a set of reactive behaviors to a hierarchical

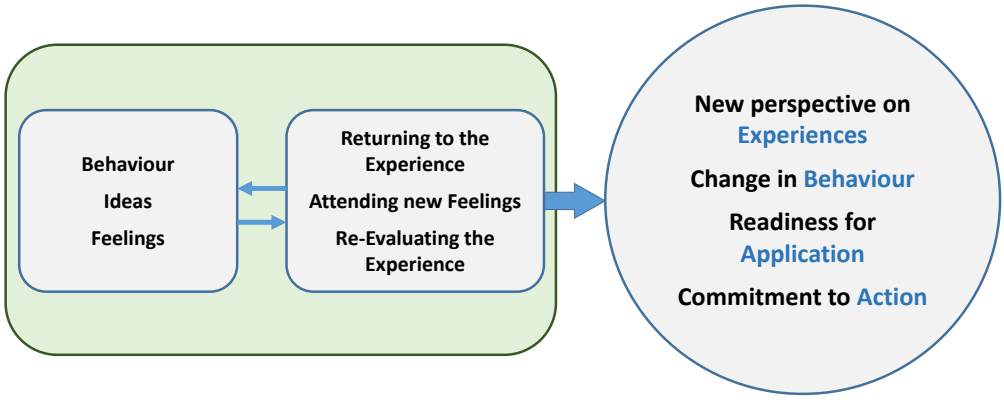


Figure 1.3. Boud's model of reflection [13].

planner [3, 26], controlling reactive layers with symbolic planners [27], or using deliberative modules as "advice" to reactive layers [111].

In this thesis, we adopt a hybrid architecture that binds reactive behaviors (i.e., situated action) to high-level planners. In doing so, we integrate semantic knowledge at each partition to generate empirically effective and adaptive robot behaviors, as well as to enable complex but natural human-robot interactions.

1.3. Prescription and Reflection

In the beginning of this chapter, we state that adaptive behaviors can be efficiently obtained through prescriptive situated approaches, but these methods do not easily scale to dynamic uncontrolled scenarios, where both long timescale goal deliberation and complex self-adaptation for model revision (i.e., reflection) are required. In fact, to adapt to the dynamic environment in our example, Sandy adopts a self-reflective behavior that updates decisions and world models for the execution of the assigned task. In this section, we explicate the notions of prescription and reflection, as well as their relation to this thesis.

Computational models adopted for decision making in autonomous agents are typically **prescriptive**, as they are based on the utility maximization and rationality assumptions of normative theories. As such, prescriptive approaches provide optimal and practical solutions that are tuned to specific situations and the needs of real decision makers (i.e., what agents should and can do). On the one hand, these solutions are mostly inflexible and often do not reflect empirical findings about human behaviors [128], dominated by cognitive limitations and biases, framing effects [141], heuristics and preferences. On the other hand, different behavioral (**descriptive**) models are available, that attempt to explain similar phenomena – see, for example, Prospect Theory [62], and specific attempts have been made to study the integration of human-robot dynamics for decision making [19]. Still, descriptive methods are rarely adopted in robotics, where the practicality and efficacy of prescriptive methods are desirable properties. In practice, solutions that are based on prescriptive approaches, such as Markov Decision Processes, are nowadays widely adopted and represent state-of-the art methodologies for autonomous intelligent

agents.

To tackle the limitations of prescriptive approaches, and to increase their effectiveness in dealing with human behaviors, **reflection** capabilities (see Figure 1.3) can be integrated within autonomous decision makers. Reflection, in fact, is an important human activity that is used to recapture experience, think about it and evaluate it [13]. In this sense, multiple functionalities in computation require reflection [84] and define a reflective computation model that is characterized as follows:

> *Reflective computation does not directly contribute to solving problems in the external domain of the system. Instead, it contributes to the internal organization of the system or to its interface to the external world. Its purpose is to guarantee the effective and smooth functioning of the object computation. (Maes and Nardi, 1988)*

To obtain a similar model, Weber and Coskunoglu [147] propose a relaxation of the rigid data requirements and assumptions of prescriptive models. This relaxation is based on the anticipation and detection of non-normative behaviors, as well as on their explication to the decision maker by means of a reflective language [44, 84] (e.g., Prolog), a knowledge base and inference mechanisms for symbolic manipulation. More recently, Zhang et al. [150] also address the problems of interpreting human actions, recognizing new situations, and making appropriate decisions. To this end, they introduce an indicator measure that allows a robot to self-reflect and reason about the novelty of observations with respect to previously learned models. This metric is then used for decision making in autonomous robots to evaluate the execution risk of the agent's actions or, from the perspective of this thesis, their equivalent affordability.

Back to our example, Sandy clearly adopts a self-reflective behavior during the execution of its assignment and while interacting with Daniele. After not finding the slotted screwdriver on the expected platform, in fact, the robot reconsiders both its world knowledge and its planned behavior. However, despite the introduction of predictive [151] and risk-sensitive [150] applications in decision frameworks, Sandy's capabilities are far beyond those of a real autonomous agent. In particular, existing decision making and planning tools still lack of integrated semantic knowledge, interaction as well as preference modeling capabilities. In this thesis, by adopting a meta-logic language to model world knowledge as proposed in [147] as well as an affordance-based action selection strategy, we leverage semantic information to generate robot behaviors through both prescriptive decision and planning frameworks. In our work, these frameworks are supported by flexible control methodologies, rich knowledge bases, and simple forms of meta-reasoning that support explanatory and interaction capabilities.

1.4. Leveraging Semantics and Interaction for Robot Policy Learning

As introduced in the previous sections, it is important for a robot to be able to use semantic information and to revise its knowledge at multiple levels of abstraction,

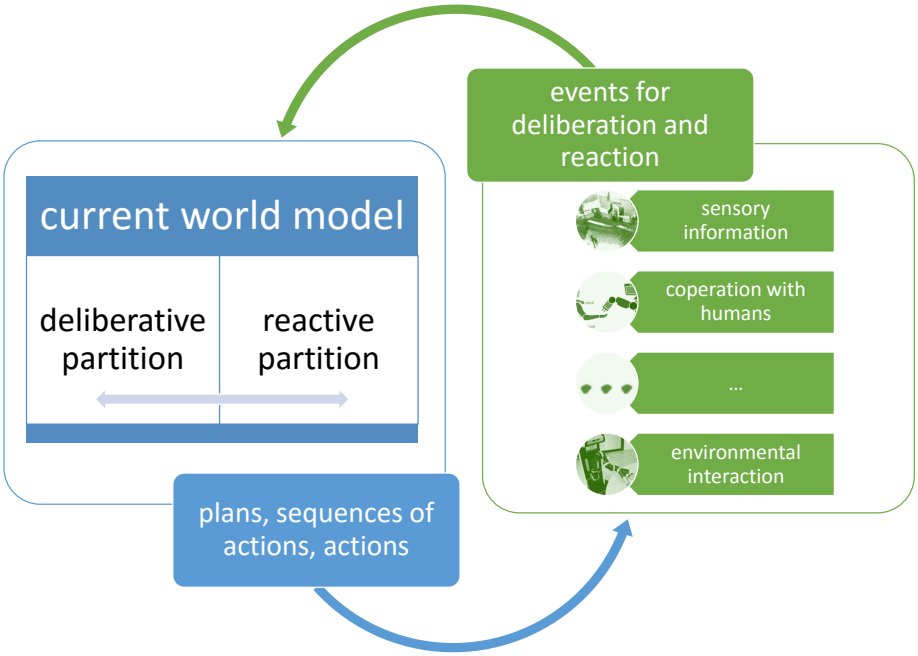


Figure 1.4. Schema of procedure for updating the world model.

coherently with a hybrid control architecture. At the deliberation level, this corresponds to using a flexible world model featuring easily upgradable, but rich abstract information. For situated action, instead, this amounts to having an explicit notion of spatio-temporal information, that integrates both the semantics and physical dynamics of the environment.

Intuitively, to capture this type information, the robot needs to actively operate within the environment and, possibly, interact with humans. In fact, on the one hand, a priori models cannot represent world knowledge appropriately, due to wrong assumptions, modeling inaccuracies and, ultimately, lack of full information; on the other hand, active interaction, both with the environment and with humans, facilitates information gathering as well as knowledge grounding and contextualization. For example, in the previously discussed scenario where our robot Sandy is assigned a pick-up task, directly operating within the environment enables the agent to autonomously collect new information about Daniele and to develop new behavioral models. With the newly generated behaviors, Sandy is able to robustly interact with its companion and accurately decide for the best action sequences.

Similarly, in this thesis we generate robot behaviors by collecting semantic information through interaction with the environment and collaboration with humans. Under this approach, illustrated in Figure 1.4, at each time-step the robot interacts with the environment by using its current world model to generate, at each abstraction layer, a plan, a sequence of actions or an action. During their execution, a set of events are separately collected for each layer – these can include, but are not limited to, cooperation with humans and observation of new sensory information. This information is finally adopted, at each partition of the world model, for updating and

amending previous knowledge as well as for generating the next time-indexed robot behavior. Importantly, during this process a priori knowledge is not completely rejected, and it can be exploited as a starting world model or, in the case of controlled domains, it can be used to mediate between the experience of the robot and the punctual knowledge of the domain expert.

While the previous introduction presents the background and underlying assumptions of our approach in its general form, later in this thesis we first introduce the models that enable this process both for deliberation and situated action, and then we detail the algorithms that implement these mechanisms at each partition of the world model.

1.5. Contributions

The main contributions of this thesis to behavior generation and to the support of semantic-driven approaches in the robotics community are the following:

- The **novel formal model of semantic mapping** extends and formalizes previous definitions [102] of semantic maps, supports knowledge revision methodologies for long-term deliberation and aims at uniforming semantic mapping solutions for knowledge representation. Such formal model includes a description of the semantic mapping process, and extends it to an incremental setting for knowledge acquisition and behavior generation. On top of this, our contribution includes a first (to the best of our knowledge) discussion of hypotheses for metrics and evaluation criteria for semantic maps, as well as the use of the model to build a dataset that is based on real sensor data.

- The **novel spatio-temporal affordance model** extends the notion of spatial semantics, connecting it to the dynamics of the environment and thus enabling semantic-driven situated behaviors. This concept builds on ecological theories (as introduced in Section 2.2), and extends previous approaches by directly connecting the environment, its spatial semantics and their evolution. Our contribution includes the formalization of a novel model that is both time-invariant and time-dependent, and that directly represents action semantics. Additionally, it defines a procedure for learning such representation, and establishes a direct connection between the notion of spatio-temporal affordances and the concept of semantic maps.

- The **online semantic knowledge acquisition schema** implements a new paradigm for model update at the deliberation level and relies both on human-robot interaction and robot experience. This schema defines a principled methodology for collecting semantic information by using our model of semantic maps. Additionally, it accounts for a task-oriented evaluation of the procedure, by using the acquired knowledge to generate behaviors via reasoning and inference.

- The **affordance-based situated action and control paradigm** integrates standard decision making methodologies with the notion of spatio-temporal affordances, as well as implements semantic-driven behaviors during situated

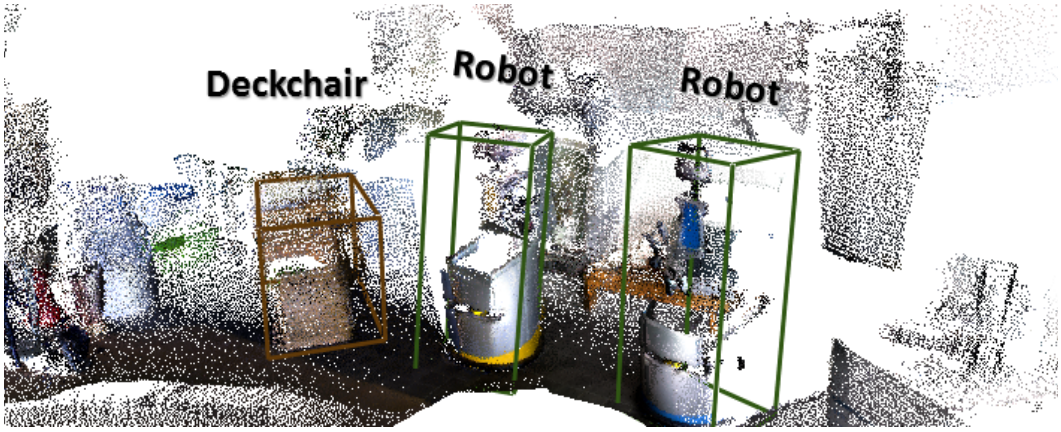


Figure 1.5. Example from the benchmarking dataset for semantic maps: view of the Robot Innovation Facility of Peccioli.

action for robot control. In particular, the paradigm leverages Monte Carlo tree search and data aggregation (see Section 2.4.2 and 3.3) to learn a policy by means of our model of spatio-temporal affordances. Our approach defines and evaluates a novel procedure for learning both time-invariant and time-dependent spatio-affordances in the context of Markov decision processes.

- The **affordance learning model and the system dynamics prediction model** connect a learning framework to the affordance-based situated action and control paradigm, and implement our knowledge update schema at a reactive level. The system dynamics prediction model, in particular, enhances previous approaches to obtain accurate simulations of dynamics models that can be used for generating accurate control policies.

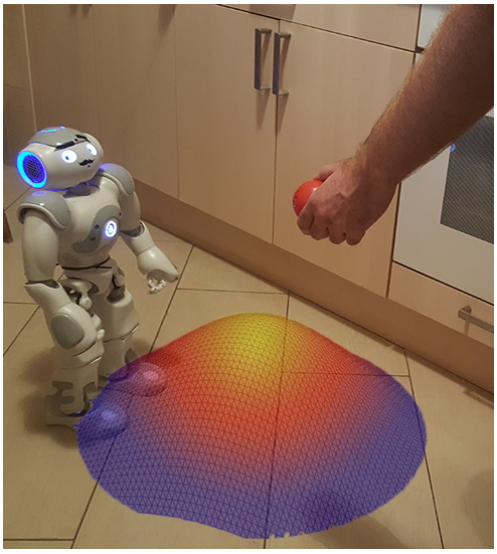
1.6. Prototype Systems and Empirical Evaluations

We empirically validate our approach by evaluating it on a number of problems and tasks for service robots. These robots are deployed in interactive and social domains, such as offices and domestic environments. To this end, we develop prototype applications that are useful for research purposes, although not yet ready for the market.

In our first application, we adopt the **formal model of semantic map** – that we develop in this thesis – to manually annotate environmental information coming from sensor data. We employ the generated models to construct a benchmarking dataset for semantic maps (Figure 1.5), thus paving the way to comparative evaluations between different approaches. In our second prototype application, we instead adopt our model of semantic mapping for enabling robots to acquire knowledge and execute tasks through interactive behaviors.

We use our model of **spatio-temporal affordances** (Figure 1.6) for supporting a robot while learning both action semantics and situated policies. In particular, we apply our algorithms for learning affordance guided policies on a humanoid platform that performs human-robot handovers.

(a) DIAGO: one of the "social robots" on which we implemented our prototype applications.



(b) Spatio-temporal affordance map for an interactive humanoid robot.

Figure 1.6. Prototype applications using the concept of spatio-temporal affordance maps.

Finally, we test our dynamics learning methods on a number of simulated and real world benchmarking applications, including a simulated helicopter and a Baxter robot.

1.7. Thesis Organization

This thesis is organized into four parts: Preliminaries, Models, Algorithms, and Conclusions. Each part and chapter of the thesis is organized and described as follows:

Part I, Preliminaries: The main background techniques and theories that are employed in the thesis and related work for semantic mapping, affordance and policy learning in robotics.	
Chapter 2	Review of the main concepts related to semantic mapping and knowledge representation, affordance theory and ecological robotics; review of decision making models and processes; introduction of the notations and terminology adopted in the thesis.
Chapter 3	Critical analysis and review of the literature related to semantic mapping, reasoning and affordance-based approaches in robotics; review of imitation learning, reinforcement learning and optimal control theories, dataset aggregation methodologies and inverse problems for reinforcement learning; analysis of relations among the approaches in literature and discussion of their limitations.

Part II, Models: Formalization of semantic mapping and spatio-temporal affordance map models.	
Chapter 4	Introduction of an easily extensible formalism for representing knowledge in semantic maps and definition of the semantic mapping process. Use of the formalism for the generation of a semantic mapping dataset.
Chapter 5	Notion of spatio-temporal affordance maps, formalization and general schema of a spatio-temporal affordance map.
Part III, Algorithms: Online semantic mapping, policy learning with spatio-temporal affordance maps, dynamics learning and random feature inverse reinforcement learning.	
Chapter 6	Semantic mapping algorithm for online knowledge acquisition, update and use based on smooth and multi-modal human-robot interactions. The approach is validated by means of a thought experiment and on service robots deployed in office and domestic scenario.
Chapter 7	Policy learning algorithms for robot control, that leverage the use of continuously refined spatio-temporal affordance maps, Monte Carlo tree search and data aggregation.
Chapter 8	Review of system identification and predictive state representation theory; data aggregation and predictive state representation algorithms for learning dynamics models in model-based reinforcement learning.
Part IV, Conclusions: Brief discussion of open research problems and concluding remarks.	
Chapter 9	Summary, conclusions and final remarks of the thesis. Open research opportunities, possible extensions and applications of the thesis work, limitations, and critical discussion.

1.8. Related Publications

Portions of this thesis have previously appeared as the following workshop, conference and journal publications:

2016

- Riccio, F., Capobianco, R., and Nardi, D. *Learning Human-Robot Handovers Through π-STAM: Policy Improvement With Spatio-Temporal Affordance Maps.* In Proceedings of the IEEE-RAS International Conference on Humanoid Robots (2016).

- Riccio, F., Capobianco, R., and Nardi, D. *Using spatio-temporal affordances to represent robot action semantics.* Workshop on Machine Learning Methods for High-Level Cognitive Capabilities in Robotics at IROS (2016).

- Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D., and Bagnell, J. A. *Improved Learning of Dynamics Models for Control.* International Symposium on Experimental Robotics, ISER (2016).

- Riccio, F., Capobianco, R., and Nardi, D. *Using Monte Carlo Search With Data Aggregation to Improve Robot Soccer Policies.* Proceedings of the 20th International RoboCup Symposium (2016).

- Sun, W., Capobianco, R., Gordon, G. J., Bagnell, J. A., and Boots, B. *Learning to smooth with bidirectional predictive state inference machines.* Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence, UAI (2016).

- Riccio, F., Capobianco, R., Hanheide, M., and Nardi, D. STAM: *A framework for Spatio-Temporal Affordance Maps.* Proceedings of the 2016 Workshop on Modelling and Simulation for Autonomous Systems, MESAS (2016).

- Capobianco, R., Gemignani, G., Iocchi, L., Nardi, D., Riccio, F., and Nardi, D. *Contexts for Symbiotic Autonomy: Semantic Mapping, Task Teaching and Social Robotics.* AAAI Symbiotic Cognitive Systems Workshop (2016).

- Gemignani, G., Capobianco, R., Bastianelli, E., Bloisi, D. D., Iocchi, L., and Daniele Nardi *Living with Robots: Interactive Environmental Knowledge Acquisition.* Robotics and Autonomous Systems (2016).

2015

- Capobianco, R., Serafin, J., Dichtl, J., Grisetti, G., Iocchi, L., and Nardi, D. *A proposal for semantic map representation and evaluation.* In Proceedings of the 7th European Conference on Mobile Robots, ECMR (2015).

- Gemignani, G., Capobianco, R., and Nardi, D. *Approaching qualitative spatial reasoning about distances and directions in robotics.* AI*IA (2015).

2014

- Capobianco, R. *Robust and incremental robot learning by imitation.* In AI*IA Doctoral Consortium (2014).

- Capobianco, R., Gemignani, G., Bloisi, D. D., Nardi, D., and Iocchi, L. *Automatic extraction of structural representations of environments.* In Proceedings of the 13th International Conference, IAS-13 (2014).

- Gemignani, G., Nardi, D., Bloisi, D. D., Capobianco, R., and Iocchi, L. *Interactive semantic mapping: Experimental evaluation.* In Proceedings of the International Symposium on Experimental Robotics, ISER (2014).

- Capobianco, R., Gemignani, G., Nardi, D., Domenico Bloisi, and Iocchi, L. *Knowledge-based reasoning on semantic maps.* AAAI Spring Symposium (2014).

2013

- Bastianelli, E., Bloisi, D. D., Capobianco, R., Cossu, F., Gemignani, G., Iocchi, L., and Nardi, D. *On-line semantic mapping.* Proceedings of the 16th International Conference on Advanced Robotics (ICAR), (2013).

- Bastianelli, E., Bloisi D., Capobianco, R., Gemignani, G., Iocchi, L., and Nardi, D. *Knowledge representation for robots through human-robot interaction.* CoRR, abs/1307.7351 (2013).

Part I

Preliminaries

Chapter 2

Background

Whatever is a reality today, whatever you touch and believe in and that seems real for you today, is going to be – like the reality of yesterday – an illusion tomorrow

— Luigi Pirandello

The idea of generating semantic-driven robot behaviors originates from existing concepts in robotics and artificial intelligence. In particular, this thesis extends and formalizes previous definitions of semantic maps, connecting them with the notion of spatio-temporal affordances and spatial semantics. This extension directly applies to existing planning and decision making frameworks, through which adaptive robot behaviors can be generated. In this chapter, we review the key elements and concepts of existing semantic map representations, and we introduce the planning and decision making approaches that we rely upon in this work. Finally, we familiarize the reader with the notations and terminology that we adopt throughout this thesis.

2.1. Elements of Semantic Mapping

The problem of learning and representing the semantics of environments based on their spatial location, geometry and appearance [75] is often referred in the literature with the term "semantic mapping" [102].

Definition 2.1. *A **semantic map** is a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes. Further knowledge about these entities, independent of the map contents, is available for reasoning in some knowledge base with an associated reasoning engine. (Nüchter and Hertzberg, 2008)*

Semantic maps should not only assign a certain number of labels or properties to relevant features of the environment (like in Goerke and Braun [47], Mozos et al. [92]), but also provide a representation of this knowledge in a form usable by the system for reasoning and behavior generation.

In this section, we review the key elements and concepts of existing semantic map representations. We then adopt these ideas and tools, and we use them throughout this thesis to develop and present our models and algorithms (Chapters 4 and 6).

Figure 2.1. Examples of occupancy grid maps, from the Radish dataset [57].

2.1.1. Grid Maps and Topological Graphs

Grid maps and topological graphs are core elements of the standard navigation stack of an autonomous robot. They are spatial representation of environments, that are respectively modeled by means of grids (Figure 2.1) or expressed in terms of their connectivity (Figure 2.2). More formally:

Definition 2.2. *An **occupancy grid map** or **grid map** is a fixed-size grid that statically models the occupancy of the environment. Each cell of the grid is a binary random variable C_i, whose state can be* free *(1) or* occupied *(0), and the map M is represented as a probability distribution*

$$p(M) = \prod_i p(C_i = 1). \tag{2.1}$$

Definition 2.3. *A **topological graph** is a concise description of the structure of the environment. It is an explicit representation of the connectivity between regions and objects, based on an abstraction of the environment in terms of places (modeled as vertices) and movements between places (modeled as edges) [33]. Such representation consists of a graph $G = (V, E)$, where V is a set of vertices and E is a set of edges among them.*

These representation are generally used in semantic maps to associate a spatial reference to each semantic annotation.

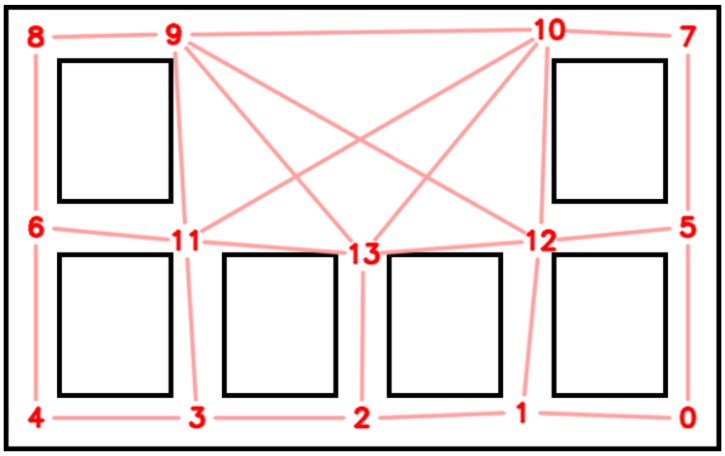


Figure 2.2. Example of a topological graph, from Khandelwal and Stone [66].

2.1.2. Knowledge Bases, Terminological and Assertion Components

Annotations and labels that belong to a semantic map are typically stored in a Knowledge Base (KB). A knowledge base is used by a robot both for reasoning and planning, by means of a reasoning engine.

Definition 2.4. *A **knowledge base** is a set of sentences, each of them expressed in a certain knowledge representation language, that represent some assertions about the world [124].*

Definition 2.5. *A **reasoning engine** is a software that uses information stored in a knowledge base, infers logical consequences, and draws new conclusions.*

Often, a knowledge base for semantic maps is represented in Description Logics (DL) [96] or in Prolog, and it is composed by two elements that are called terminological box (Tbox) and assertion box (Abox) [29].

Definition 2.6. *The **TBox** stores a set of universally quantified assertions, stating general properties of concepts and roles [29].*

Definition 2.7. *The **ABox** contains TBox compliant assertions on individual objects [29], such as instances of certain concepts.*

Several reasoning tasks can be performed on a DL knowledge base, such as computing the subsumption relation between two concepts or checking logical implications of certain assertions.

2.1.3. The Symbol Grounding Problem

One of the main challenges of semantic mapping consists in tackling the symbol grounding problem [53], that is the issue of associating spatial representations, as introduced in Section 2.1.1, with symbolic annotations stored in the knowledge base of a semantic map (Section 2.1.2).

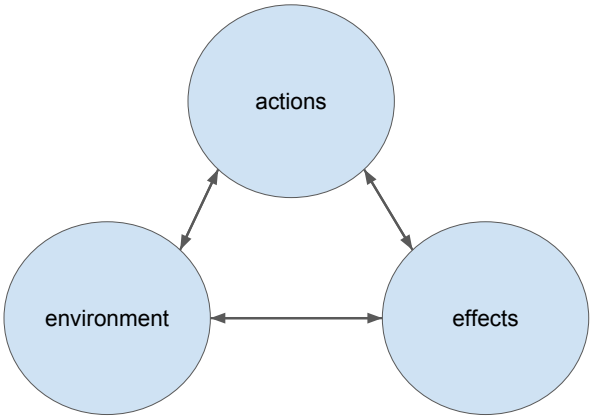


Figure 2.3. Ecological model adopted in this thesis.

Definition 2.8. *The **symbol grounding problem** consists of the interpretation of a formal symbol system and its connection to anything that is non-symbolic, such as sensory representations.*

Multiple approaches have been proposed to perform symbol grounding and to represent semantic maps. In this thesis, we review some of them in Section 3.1.

2.2. Ecological Robotics and Affordances

Recently, the relationship between the agent and its environment has been used by researchers to build complex and flexible representations of behaviors, that are able to better address the dynamics of the world. These representations are named affordances and, by directly modeling the interaction of an agent with its own environment, they leverage the theories of ecological psychology [56]. Ecological theories, in fact, consider animals and their environments as "inseparable pairs", that have to be described according to the behavior of the animal itself [32]. The extension of this perspective to the robot domain defines an alternative paradigm of ecological robotics.

Definition 2.9. *Under the paradigm of **ecological robotics**, a robot and its environment are considered as a unique system, in which the environment provides the information for the adaptation of the agent's behavior and the dynamics of the world directly affects action choices.*

Affordances, in particular, have been originally introduced by Gibson [46] as action opportunities that objects offer.

Definition 2.10. *The **affordances** of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. An affordance is equally a fact of the environment and a fact of behavior. It is both physical and psychical, yet neither. An affordance points both ways, to the environment and to the observer. (Gibson, 1979)*

Alternative models exist, that consider affordances as relations between animals and features of situations [23], or where action opportunities depend on the experience

of the agent [101]. In this work, we adopt a definition of affordances (Figure 2.3) that depends on the experience of the agent and whose shape, or distribution, is modified by events that occur in the environment at each time, as effects of the agent's actions.

Definition 2.11. *An affordance is a value function that represents the desirability of an action over a full environment, given its state at a certain time.*

While we present existing literature on affordances (used in robotics) in Section 3.2, we use this notion in Chapters 5 and 7.

2.3. Petri Net Plans

In this section, we introduce the concept of Petri Net Plans (PNPs), that we leverage in this thesis (Chapter 6) for executing our long-term sequence of actions and, hence, for implementing our deliberated plans.

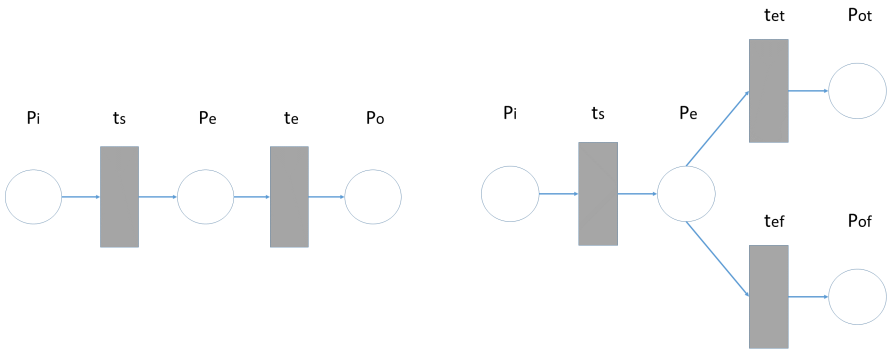
Petri Net Plans (PNP) [154] is a rich formalism for plan representation, based on Petri Nets [93] that can be used to express non-instantaneous actions, sensing and conditional actions, action failures, concurrent actions, interrupts, and action synchronization (in a multi-agent context). More formally:

Definition 2.12. *A **Petri Net** is a directed bipartite graph, in which nodes represent either **transitions** (events, represented by bars) or **places** (conditions, represented by circles). The directed arcs describe which places are pre-conditions (inputs) or post-conditions (outputs) for each transition.*

Definition 2.13. ***Petri net plans** are Petri Nets of a restricted form, where:*

- *input places (p_i) model the initial configurations of the network before the action has been executed;*
- *execution places (p_e) model the configurations during which the action is executed. These places are also used to create hierarchical plans by calling a sub-plan while visiting this place;*
- *output places (p_o) model the final configurations of the network after the execution of the action;*
- *start transitions (t_s) model the events that trigger the execution of the action;*
- *end transitions (t_e) model the events that trigger the end of the action. In a sensing action there are a true (t_{e_t}) and a false (t_{e_f}) end transitions that fire depending on the outcome of the query made to the knowledge base.*

Under this formalism, plans are composed of basic actions – i.e., specific combinations of places and transitions – that can be combined through a set of operators for executing complex tasks. These actions can be ordinary (Figure 2.4a) or sensing actions (Figure 2.4b). The former have a certain duration, while the latter depend on one or more conditions, that are verified at run-time by querying an external



(a) PNP ordinary action. **(b)** PNP sensing action.

Figure 2.4. Ordinary and sensing actions in a Petri Net Plan.

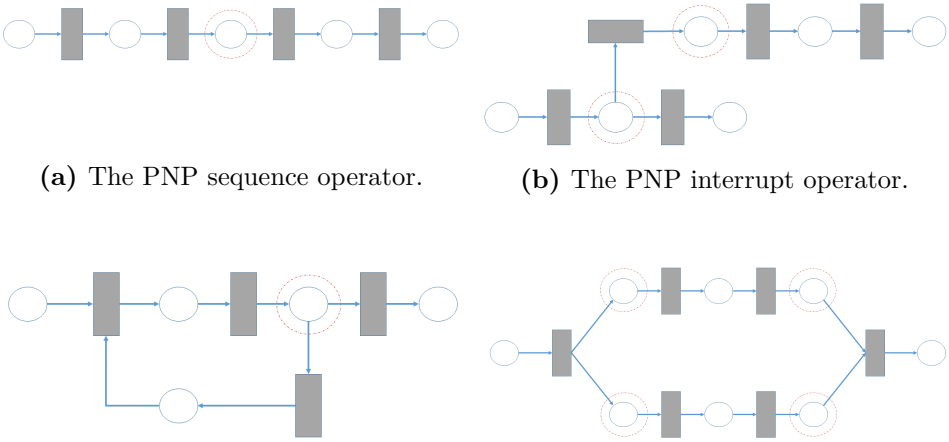
knowledge base. No restriction is imposed on the knowledge base, which is supposed to be updated by other modules according to the perceptions of the agent.

Basic actions can be composed by using a set of operators for single and multi-agent plans. Here we recall some of the single-agent operators (Figure 2.5), referring the reader to Ziparo et al. [154] for a formal and more complete definition of the available operators:

- *sequence*: a sequence is obtained by merging two places of different PNPs. It can be applied to any place except for the execution ones, as shown in Figure 2.5a;
- *interrupt*: an interrupt connects the execution place of an action (or sub-plan) to a non-execution place of another network. It causes the temporary termination of the action (or sub-plan) under anomalous conditions, as shown in Figure 2.5b. Interrupts are also used to repeat a portion of a plan that do not achieve its post-conditions in order to implement while-loops;
- *loop*: loops enable the repetition of a portion of the plan until a certain event occurs (*do-until*), or if a set of actions needs to be repeated a given number of times (*do-n-times*) (Figure 2.5c).
- *fork* and *join*: each token in a Petri Net Plan can be considered a thread in execution. Through fork and join, threads can be created and synchronized. An example of a fork followed by a join is shown in Figure 2.5d.

2.4. Models of Decision Processes

Multiple approaches have been adopted for implementing decision making in autonomous agents, with the goal of obtaining mathematical frameworks to represent decision problems and then infer optimal solutions. In this section, we review the Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs), prescriptive frameworks that aim at maximizing the expected utility of a sequence of interactions with a stochastic process.



(a) The PNP sequence operator.

(b) The PNP interrupt operator.

(c) The PNP loop operator. It can be used to define definite or indefinite loops.

(d) The PNP fork and join operators.

Figure 2.5. Operators in a Petri Net Plan.

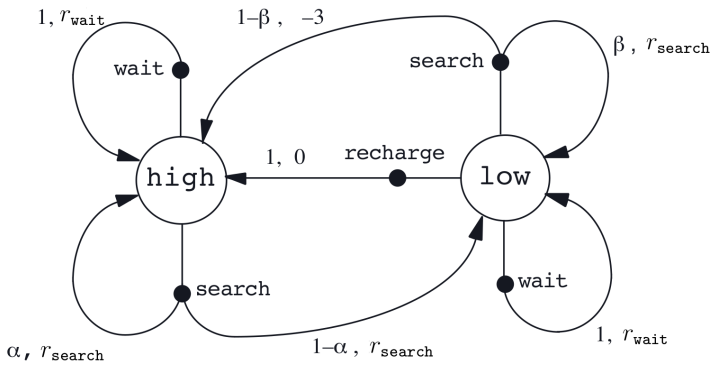


Figure 2.6. Examples of Markov Decision Process as a graph, from Sutton and Barto [138].

2.4.1. Markov Decision Processes

The Markov Decision Process (MDP) is a prominent model for planning and decision making in discrete settings. Under this framework, which we also adopt in this thesis (Chapters 7 and 8), the set of states S and actions A of an agent are described in terms of graph nodes and edges (Figure 2.6). Each pair (s, a), composed by a node s and an arc a, has an associated reward $r_{s,a}$ and the transition between states can be deterministic or stochastic. More formally:

Definition 2.14. *A **Markov decision process** is a tuple*

$$\mathcal{M} = (S, A, T, R, \gamma),$$

where:

- $\mathcal{S}$ *is the set of states of the environment;*
- $\mathcal{A}$ *represents the set of actions;*

- $T : S \times A \times S \to [0, 1]$ *is the transition function that models the probabilities of transitioning from state* $s \in \mathcal{S}$ *to* $s' \in \mathcal{S}$ *when taking action* $a \in \mathcal{A}$*;*
- $R : S \times A \to \mathbb{R}$ *is the reward function;*
- $\gamma \in (0, 1]$ *is a discount factor.*

In this setting, transitions and rewards are assumed to be Markovian – i.e., a function of the current state only, and decisions are represented through a policy π, that defines the behavior of an agent by mapping states to actions. Policies can be classified as **deterministic** or **stochastic**. A deterministic policy $\pi(s)$ specifies a unique action based on the current state, while stochastic policies choose actions according to a probability distribution $\pi(a|s) \in [0, 1]$. More classes of policies exist, that can be categorized as **non-stationary** or **stationary** depending on whether they respectively maintain a notion of time or not.

By executing a policy, an agent interacts with its environment in discrete time-steps and defines a **sequence** (or a **trajectory**) of state-action pairs $\zeta = (s_t, a_t)$, with $t = 0, \dots, T$, and an associated **cumulative reward** $R(\zeta) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$. In the case of deterministic agents (i.e., agents with deterministic policies), the goal consists in finding a policy $\pi(s)$ that maximizes its **expected cumulative reward** $\mathbb{E}_{\zeta \sim \pi}[R(\zeta)]$ over a finite or infinite time horizon T. This can be obtained by using the notions of **state-value function** $V^\pi(s)$ (Eq. 2.2) and **action-value function** $Q^\pi(s, a)$ (Eq. 2.3) of a policy

$$V^\pi(s) = \sum_a \pi(a|s)\{R(s, a) + \gamma \sum_{s'} T(s, a, s')V^\pi(s')\} \tag{2.2}$$

$$Q^\pi(s, a) = R(s, a) + \gamma \sum_{s'} T(s, a, s')V^\pi(s'). \tag{2.3}$$

Intuitively, the state-value function corresponds to the value of the expected return when starting in state s and following policy π from there, while the action-value function represents the value of the expected return when taking action a in state s and then following policy π.

By using these concepts and solving their corresponding Bellman optimality equations [138]

$$V^*(s) = \max_a \{R(s, a) + \gamma \sum_{s'} T(s, a, s')V^*(s')\} = \max_a Q^*(s, a) \tag{2.4}$$

$$\begin{aligned} Q^*(s, a) &= R(s, a) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q^*(s', a') \\ &= R(s, a) + \gamma \sum_{s'} T(s, a, s')V^*(s') \end{aligned} \tag{2.5}$$

an optimal policy can be greedily determined with one look-ahead, as in Eq. 2.6, or just by choosing the best action according the optimal action-value function 2.7.

$$\pi^*(s) = \arg\max_a \{R(s, a) + \gamma \sum_{s'} T(s, a, s')V^*(s')\} \tag{2.6}$$

$$\pi^*(s) = \arg\max_a \{Q^*(s, a)\}. \tag{2.7}$$

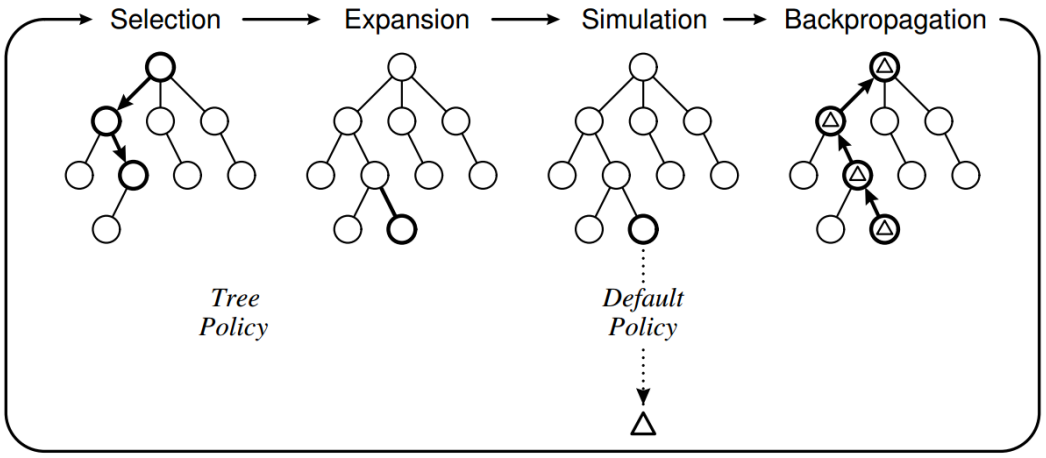


Figure 2.7. Steps of one iteration of Monte Carlo Tree Search, from Browne et al. [15]

In fact, the **policy improvement theorem** establishes a criterion for comparing policies and, hence, for defining optimality. Note that, although this theorem explicitly considers deterministic policies, similar principles apply in the case of stochastic policies.

Theorem 2.1. *Given any pair of deterministic policies π' and π, if*

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$$

holds for all $s \in S$, then π' is at least as good as, π. Hence, for all states $s \in S$, π' must obtain an expected return greater than or equal to π:

$$V^{\pi'}(s) \geq V^{\pi}(s).$$

Hence, whenever the relation $Q^{\pi}(s, a) \geq V^{\pi}(s)$ holds, it is better to change π so that action a is chosen in state s. Based on this, a simple mechanism of **policy iteration** can be developed, that interleaves between policy evaluation (Eq. 2.6) and policy improvement (Eq. 2.6) until convergence. Other approaches, instead, perform **value iteration** by directly updating $V^*(s)$ through Eq. 2.4 and 2.5.

2.4.2. Monte Carlo Tree Search

As we explained in the previous section, either value iteration or policy iteration can be used to solve MDPs and obtain optimal policies. However, these approaches are often unfeasible in domains characterized by very large state spaces, such as multi-agent systems. This is due, for example, to complex dynamics and costly computations for full state space explorations, including portions of it that are "non interesting" for the considered agent. For this reason, tractable algorithms have been developed, that approximate state-action values by collecting random samples in the decision space and by building a search tree according to the results. Among them, **Monte Carlo Tree Search** (MCTS) is a statistical algorithm that can be used with little or no domain knowledge [15], and that recently led to notable successes [133].

MCTS evolves through multiple iterations, each of which is composed of four fundamental steps – illustrated in Figure 2.7 – that build a search tree to support action planning:

- *selection*: a selection policy is used to reach a leaf node from the root of the current tree – i.e., in each node, starting from the root, an action is applied according to the selection policy;
- *expansion*: when the leaf node is non-terminal, the leaf is expanded by creating new child nodes – according to the available (or possible) actions – and selecting one of them;
- *simulation*: a default policy (e.g., random) is adopted starting from the new node until an outcome is achieved or termination is reached;
- *back-propagation*: each traversed node in the tree updates its statistics according to the achieved outcome. To evaluate the statistical significance of the outcome, more simulations are typically carried out before back-propagation.

As described, MCTS is achieved by means of two policies: a selection (or tree) policy, that is used within the tree, and a default policy, used to carry out simulation. Intuitively, iteration after iteration, the algorithm builds a partial tree that is used to estimate action values through simulation. These estimates improve with the number of iterations, and they are used to guide the construction of the tree itself. While further details can be found [15], in this thesis (Chapter 7) we leverage a variant of MCTS that is named **Upper Confidence Bounds for Trees** (UCT).

UCT models the node selection step of the algorithm as a multi-armed bandit problem. To this end, the algorithm maintains a count of its visits to each state and action, and adopts a simple upper confidence bound strategy – UCB1 [5] – to balance between exploration and exploitation on the tree. In particular, at each iteration $h = 1 \dots H$, the algorithm simulates the execution of each legal action in s_h and selects the best action a_h^* as

$$e = C \cdot \sqrt{\frac{\log(\sum_a n(s_h, a))}{n(s_h, a)}} \tag{2.8}$$

$$a_h^* = \arg\max_a Q(s_h, a) + e, \tag{2.9}$$

where $Q(s_h, a)$ is the action value function, C is a constant that multiplies and controls the exploration term e, and $n(s_h, a)$ is the number of occurrences of a in s_h. The action value function $Q(s, a)$ is obtained by back-propagating the final reward of each simulation to all the traversed states s.

2.4.3. Linear-Quadratic Regulators

The Linear-Quadratic Regulator (LQR) is a special case of MDP characterized by continuous states and actions, linear state dynamics and quadratic cost functions. More formally:

Definition 2.15. *A **linear-quadratic regulator** is a special MDP, represented as a tuple*

$$\mathcal{L} = (\mathcal{S}, \mathcal{A}, A, B, \Sigma, Q, R),$$

where:

- $\mathcal{S} = \mathbb{R}^m$ *is the continuous state space;*
- $\mathcal{A} = \mathbb{R}^n$ *is the continuous action space;*
- $A \in \mathbb{R}^{m \times m}$ *and* $B \in \mathbb{R}^{m \times n}$ *are linear matrices that define the linear transition function* $s_{t+1} = As_t + Ba_t + \nu_t$*, while* $\nu \sim \mathcal{N}(0, \Sigma)$ *is Gaussian noise;*
- Q *and* R *are positive semi-definite matrices that parametrize the reward function as* $-s_t^\top Q s_t - a_t^\top R a_t$*.*

For this class of problems, a solution can be efficiently computed by defining a total cost J - or a value function - accumulated over a certain horizon T. When the Gaussian noise ν is not present – i.e., the transition function is deterministic, at a time-step t the optimal cost-to-go $J = s^\top P_t s$ is obtained through the optimal policy $\pi^*(s) = -K_t s$, where K_t is defined in terms of the solution $P_t \in \mathbb{R}^{m \times m}$ to the discrete-time Riccati difference equation:

$$K_t = -(R + B^\top P_{t-1} B)^{-1} B^\top P_{t-1} A \tag{2.10}$$

$$P_t = Q + K_t^\top R K_t + (A + BK_t)^\top P_{t-1}(A + BK_t). \tag{2.11}$$

This procedure corresponds, in all respects, to a sequence of value iteration updates starting from $P_0 = 0$ and performed in closed form. Similar solutions can be derived in presence of ν – i.e., when the transition function is stochastic. Finally, this approach can also be adopted in the case of non-linear systems by simply computing, around the trajectory generated by the current policy, a linear approximation of the transition function and a quadratic approximation of the reward function. In this thesis, we use linear-quadratic regulators for some of our experiments in Chapter 8.

2.4.4. Partial Observability

The Partially Observable Markov Decision Process (POMDP) is an extension of the MDP where the states are not fully accessible. Under this common setting, in fact, the agent only receives some state-dependent observations, rather than the true states, while the underlying states, transitions and rewards functions remain unchanged from the traditional MDP. More formally:

Definition 2.16. *A **partially observable Markov decision process** is a tuple*

$$\mathcal{P} = (S, A, O, T, R, Z, \gamma),$$

where:

- $\mathcal{S}$ *is the set of states of the environment;*

- $\mathcal{A}$ *represents the set of actions;*
- $\mathcal{O}$ *is a set of observations;*
- $T : S \times A \times S \to [0, 1]$ *is the transition function that models the probabilities of transitioning from state* $s \in \mathcal{S}$ *to* $s' \in \mathcal{S}$ *when taking action* $a \in \mathcal{A}$*;*
- $R : S \times A \to \mathbb{R}$ *is the reward function;*
- $Z : S \times O \to [0, 1]$ *is the observation dynamics that models the probability of observing* $o \in \mathcal{O}$ *from state* $s \in \mathcal{S}$*;*
- $\gamma \in (0, 1]$ *is a discount factor.*

Since the agent does not directly observe the state of the environment, this framework is connected to the notions of **belief state** and **belief space**. A belief state $b(s) = p(s)$ is a probability distribution over the states, while the belief space the is the set of all possible probability distributions – i.e., the available probability space. At each time, the agent updates its belief according to

$$b'(s) = \eta Z(s, o) \sum_{s'} T(s, a, s')b(s'), \tag{2.12}$$

where $b'(s) = \sum_{s'} T(s, a, s')b(s')$ is the action update rule and $b'(s) = \eta Z(s, o)b(s)$ is the observation update rule, with η normalization factor.

2.5. Notation and Terminology

In this section we describe the notation and terminology that we adopt throughout this thesis. In particular, we adhere to the following notation, where each x, or x_i, is a variable:

- **Sets of values:** $\{x_i\} = \{x_1, \ldots, x_n\}$;
- **Vectors of values:** $\mathbf{x} = (x_1, \ldots, x_n)^\top$;
- **Probability distribution:** $p(X = x) = p(x)$
- **Conditional probability distribution:** $p(Y = y \mid X = x) = p(y \mid x)$;
- **Expectation of a function** $f(x)$: $\mathbb{E}[f(x)] = \sum_x p(x)f(x)$;

Additionally, we refer to the following dictionary of terminologies:

- **Metric map:** occupancy grid map, as introduced in Definition 2.2;
- **System dynamics:** transition function $T(s, a, s')$ as defined in the Markov Decision Process framework;
- **i.i.d.:** independent and identically distributed.

Chapter 3

Related Work

Consider your origins: you were not made to live as brutes, but to follow virtue and knowledge.

— Dante Alighieri

In this chapter, we review existing methods for learning semantic knowledge and generating robot behaviors, under a variety of assumptions and with multiple techniques. These approaches have been grouped in three different sections: Semantic Knowledge and Robot Reasoning (Section 3.1), Affordance-Based Methods in Robotics (Section 3.2), and Approaches to Policy Learning (Section 3.3), reflecting the three key concepts adopted in this thesis. We discuss the relation between these concepts as well as our models and algorithms both in Section 3.4 and throughout the following chapters of this thesis.

3.1. Semantic Knowledge and Robot Reasoning

The development of semantic-based methodologies in robotics has been an active research area in the last few years. After Kuipers [77], who described spatial information as essential to commonsense knowledge, many authors focused on the idea of environmental semantics for robots. Such increasing interest is motivated by the need of designing cognitive robots, that are capable of acting and collaborating by understanding the environment in which humans live, as well as the way they operate in it. Nevertheless, the ability to communicate represents a strict requirement for collaboration between two or more agents. When dealing with humans, this can be naturally achieved by enabling robots to use spoken language, based on the learned semantics of the world.

In this section we review the approaches to semantic knowledge representation, acquisition, and inference that have been adopted in robotics. In particular, while defining a general taxonomy of these methods, we mostly focus on representation and knowledge acquisition issues, that help us to correctly collocate our work and to highlight its main similarities and differences with respect to previous literature.

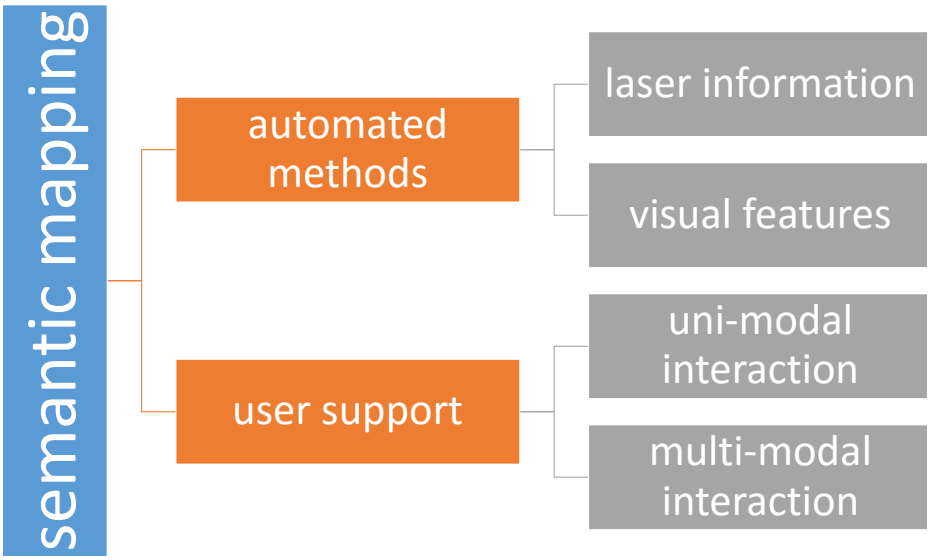


Figure 3.1. Taxonomy of approaches for semantic mapping.

3.1.1. Semantic Mapping

There exists a large literature on the problem of learning and representing the semantics of environments based on their spatial location, geometry and appearance [75]. This activity is usually referred to as "semantic mapping". Such a term, although originally describing a difficult process that deals with heterogeneous information (i.e., not limited to spatial knowledge), has strong implications. Semantic maps should, in fact, not only assign a certain number of labels or properties to relevant features of the environment, but also provide a representation of this knowledge in a form usable by a robotic system. Indeed, semantic mapping is the incremental process of mapping relevant information of the world (i.e., spatial information, temporal events, agents and actions) to a formal description supported by a reasoning engine [21, 102], with the aim of learning to understand, collaborate and communicate.

Ongoing research on semantic mapping typically considers autonomous agents endowed with navigation and manipulation skills, without specific assumptions on communicative or collaborative behaviors. Based on this, several approaches have been proposed, that can be grouped in two main categories:

- fully automated methods for the classification of locations and objects [41, 16, 39, 47, 148, 12, 92, 51];
- techniques that exploit the support of the user in the knowledge acquisition and learning process [76, 149, 108, 99, 54, 109, 113, 145].

Automated methods can be further specialized into two smaller groups. A first group of methods relies upon laser-based metric maps, that are automatically segmented and labeled through classification and clustering [41]. Very often, in these approaches, topological maps are extracted for robot navigation [16, 39, 47].

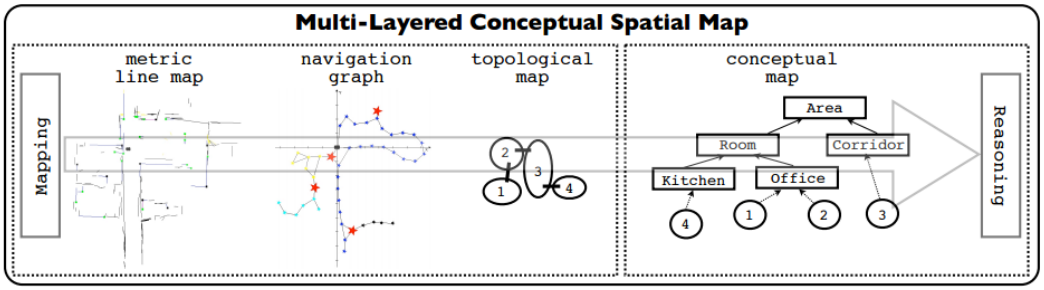


Figure 3.2. Multi-layered semantic map representation, from Zender et al. [149].

The second set of techniques, instead, uses visual features for object recognition and place categorization [148], or a combination of visual and range information provided by RGB-D cameras [92]. Significant progress has been made in this kind of automated semantic mapping [51], although robustness and generalization still represent the main difficulties of these approaches.

In human augmented mapping, users play a central role in facilitating symbol grounding, while objects and spaces are still autonomously recognized by the robot. The interaction between humans and robots is generally uni-modal, and it is achieved through spoken language. For example, Zender et al. [149] generate conceptual representations of indoor environments by merging a priori knowledge and sensor information, while the user supports the acquisition process through place labeling. On the one hand, this method has been followed by multiple authors, by adopting mixed initiative strategies [108] or probabilistic knowledge representations [99]; on the other hand, the degree of commitment required from the user varies greatly among different approaches. Pronobis and Jensfelt [109], for example, use a multi-layered mapping approach that collects semantic information and integrates the user input, whenever provided, only as additional properties about existing objects. Conversely, Walter et al. [145] fully rely on natural language descriptions to generate human-centric models of their environments. Few alternative approaches adopt more complex models of collaboration, based on clarification dialogues [76], narrated guided tours [54] or rich multi-modal interactions, including speech, vision, and pointing devices [113].

Importantly, current research on semantic mapping presents an extreme heterogeneity of methodologies for representing learned maps – that prevents comparative evaluations, standard validation and evaluation procedures, and benchmarking strategies. For example Galindo et al. [41] represent environmental knowledge by anchoring sensor data to symbols of a conceptual hierarchy, based on description logic. The authors validate their approach by building their own domestic-like environment and testing the learned model through the execution of navigation commands. Zender et al. [149] generate a multi-layered representation (Figure 3.2), ranging from sensor-based maps to a conceptual abstraction (an OWL-DL ontology). Except for individual modules, their experimental evaluation is mainly qualitative. Pronobis and Jensfelt [109], instead, represent a conceptual map as a probabilistic chain graph model and evaluate their method by comparing the belief of the robot of being in a certain location against the ground truth. Pangercic et al. [105] in-

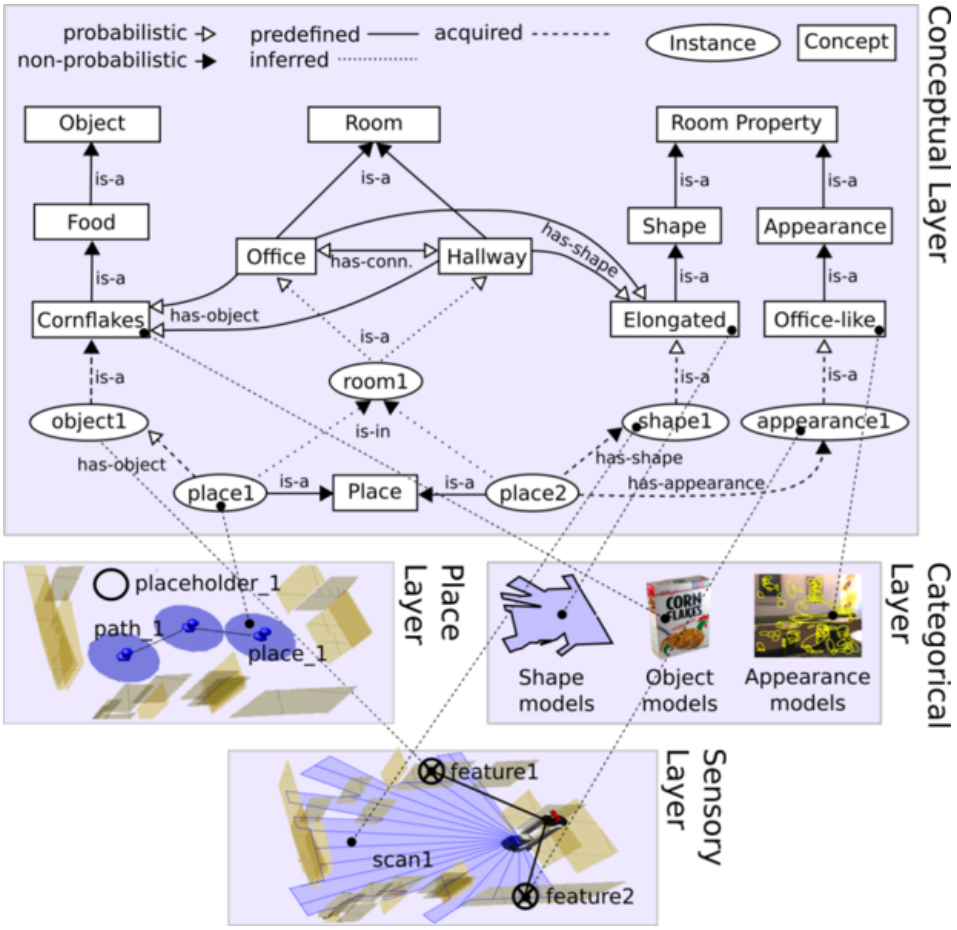


Figure 3.3. Conceptual map from Pronobis and Jensfelt [109].

vestigate the representation of "semantic object maps" (Figure 3.4) by means of a symbolic knowledge base (in description logic) associated to Prolog predicates (for inference). The authors demonstrate their approach on a PR2 robot which has to open a cabinet and to detect handles based on an a priori given semantic map. Finally, Bastianelli et al. [9] use a Prolog knowledge base containing both specific environmental knowledge and the general domain information. The knowledge base is linked to the physical environment by means of a matrix-like data structure generated on top of a metric map. Once again, the experimental validation is based on qualitative evaluations of the robot behavior, given a certain command and the learned semantic map.

In practice, none of the cited works can compare the performance of their semantic mapping method against each other. In this sense, it is important to remark that even the simplest semantic map goes far beyond "simple" labeling of spatial features. Although they are built on top of sophisticated SLAM procedures, computer vision and machine learning algorithms, semantic maps must support reasoning over the acquired knowledge. Hence, on the one hand, semantic mapping methods cannot be directly evaluated on metrics and datasets available for other algorithms, since they do not take into account any kind of reasoning capability. On

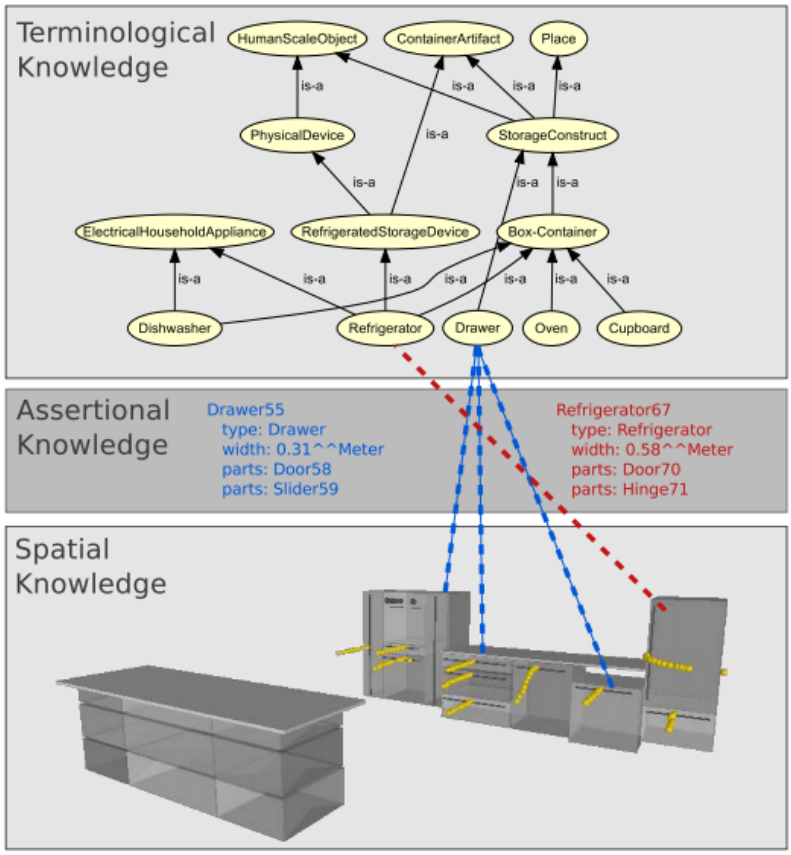


Figure 3.4. Semantic object maps, from Pangercic et al. [105].

the other hand, approaches proposed in literature lack any kind of standardization and typically underestimate these questions. In particular, two main issues emerge from the analysis of the state-of-the-art: 1) the absence of a common formalism for representing semantic maps and, consequently, 2) the lack of suitable validation and evaluation techniques.

3.1.2. Understanding Space

Spatial ability is the capacity to understand and reason about spatial relations, and it is repeated within basic activities such as navigation, horizontality perception and mental object rotations. In artificial intelligence, multiple attempts have been made to emulate human reasoning in spatial related tasks. Frank [37], for example, discusses a formal method for qualitative reasoning about distances and directions in a geographic space. While directions are expressed both through a cone and a projection based representation, his approach only yields approximate results. Zimmermann and Freksa [153], instead, use the "double cross" calculus and compare relative point positions against vectors, rather than single points. Finally, Balbiani et al. [8] adopt geometric primitives in the rectangle algebra to approximate spatial regions, at different granularities. The reasoner evaluates spatial relations considering the rectangle distribution within the environment.

Similar principles have been applied in robotics for critical tasks such as perception-based target search and self-localization. Busquets et al. [17], for example, enable multiple robots to navigate in unstructured environments through qualitative computation of landmark locations, that are categorized according to fuzzy sets of distances and angles. Klenk et al. [68] use data from a mobile robot to generate a relational representation of semantically labeled objects, showing how the boundary of a context-dependent spatial region can be defined using anchor points. Sjöö et al. [134], instead, use perceptual measures of two spatial relations, *in* and *on*, to guide a robot in a landmark-based object search. Kunze et al. [78] investigate how probabilistic models of qualitative spatial relations can improve the performance in object search tasks. Finally, Santos et al. [126] present a probabilistic algorithm for robot self-localization that is based on a topological map constructed from the observation of spatial occlusions.

While most of these works assume a complete or static knowledge about the environment, such a hypothesis typically leads to inappropriate robot behaviors in practice. Back to the example in the introduction of this thesis, in the initial world model provided to Sandy, the slotted screwdriver is located on platform A. However, due to the non-static nature of the environment this information gets invalidated by incoming sensor data. At that point, in order to complete its assigned pick-up task, the robot needs to update its model, infer new information and explore a different platform. In subsequent sections of this thesis, we contribute in enhancing robot knowledge through an incremental and online process that can revise and update information while exploiting spatial relations.

3.2. Affordance-Based Methods in Robotics

The concept of affordances was originally introduced by the American psychologist J. J. Gibson [46], who described them as action opportunities that objects offer, independently of agents' perceptual abilities. Gibson's ecological approach has been embraced both in human-computer interaction [101] and in robotics, where affordances have been mostly adopted for behavior-based control [125]. In this context, in fact, they have been used to represent [104], learn [73] and exploit [67] object related actions in dynamic scenarios. Recently, this concept has been extended to describe environments as a combination of spatial affordances for generating and supporting adaptive robot behaviors [83, 35, 117].

In this section we review the methods for representing, learning and using affordances in robotics. In particular, we make a general distinction between approaches based on object affordances and spatial affordances. This helps us to highlight the main similarities and differences of our work with previous literature.

3.2.1. Object Affordances

Affordance theory has been introduced in robotics for representing action possibilities over objects that are perceived by a robot within the environment. In particular, multiple studies focused on learning consequences of specific actions in a given situation, as well as learning object properties that afford a certain behavior.

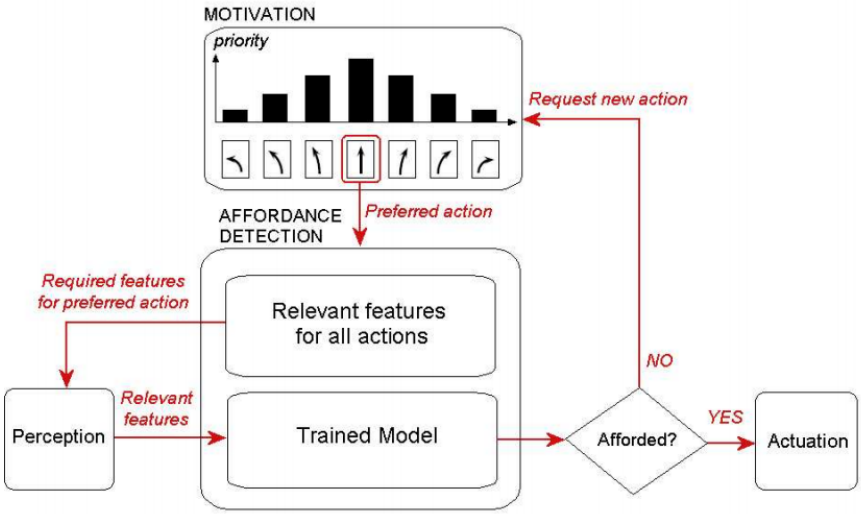


Figure 3.5. Robot control system exploiting affordances, from Ugur et al. [142].

In the case of prescriptive behaviors, consequence prediction is generally useful [147] and can be easily applied in the context of decision making and action planning. Metta [87] and Fitzpatrick [86, 36], for example, enable a robot to familiarize itself with objects through actions and to recognize other actors (such as humans) – based on their effects on learned objects. Similarly, Stoytchev [136] adopts a behavior-based approach to ground tool affordances in robot behavioral repertoires. By choosing random actions, the robot observes effects on environmental objects and exploits their expectation to solve tool-using tasks. Montesano et al. [90, 91] learn statistical relations between actions, object properties and action effects as object affordances. Based on them, they run simple imitation games that exploit both task interpretation and planning capabilities. Also, Ridge et al. [118] describe a vision platform that uses a robotic arm to interact with objects and attempts to learn affordance properties as action effects. Finally, Koppula et al. [73] extract a descriptive labeling of both sequences of human activities and associated affordances, corresponding to interactions with objects.

On a slightly different perspective, multiple studies focus on affordance discovery as a tool for improving specific robot behaviors. For instance, Murphy [94] uses three mobile robots to evaluate a methodology that isolates reliable affordances. Eventually, such affordances are exploited for executing specific robot behaviors. Cos-Aguilera et al. [28] propose a model that integrates behavior selection and affordance learning through the monitoring of internal motivations. They apply this model to an autonomous robot that interacts with objects in its environment while attempting to survive. Ugur et al. [142] analyze how affordances can affect autonomous robot control and, more specifically, how a mobile robot equipped with a 3D laser scanner can learn to perceive the traversability affordance and use it for navigation, as illustrated in Figure 3.5. On a related note, Şahin et al. [125] propose a new formalism for affordances and discuss its implications for autonomous robot control on three different tasks. Finally, Kim and Sukhatme [67] describe a

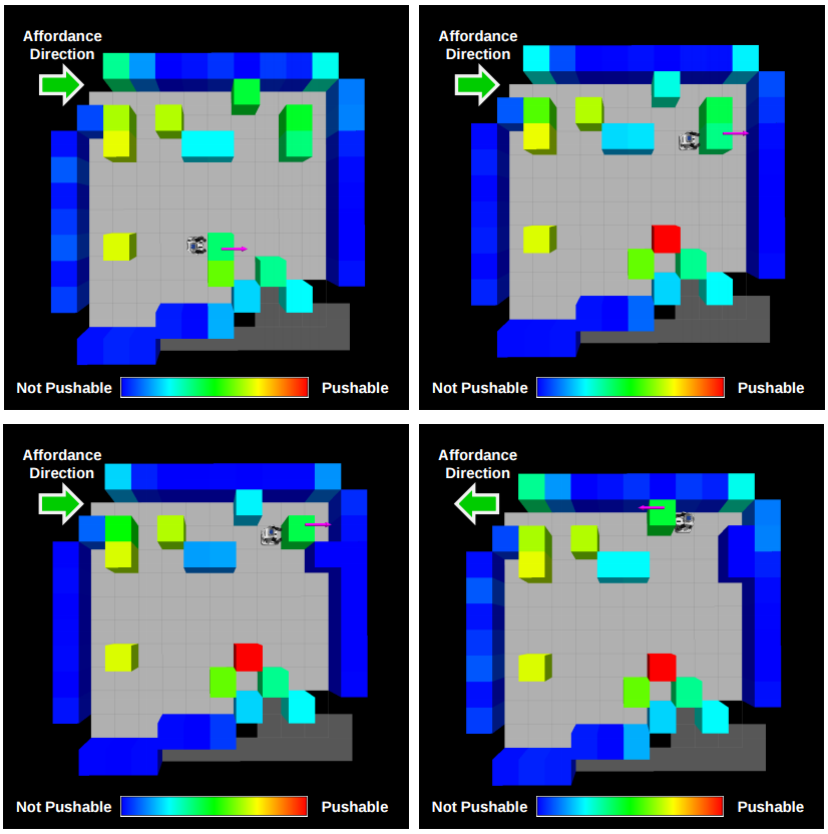


Figure 3.6. Heatmap representing the affordability of the "push" action during a sequence of robot manipulation steps in a grid world, from Kim and Sukhatme [67].

method to interactively build an affordance map for robotic tasks that is based on 2D occupancy grids and is used for interactive manipulations (see Figure 3.6).

Further work focuses on alternative applications of affordance theory. Kammer et al. [63] introduce "situated affordances" – to consider the environmental context in which an object is embedded. Pandey and Alami [104], instead, enrich the notion of affordance by incorporating agent-agent action opportunities. They consider agent affordances as an important aspect for day-to-day interaction and decision making in effort based and shared cooperative planning. Unfortunately, despite the interconnection between space and affordances in some of the previously cited work [142, 67], none of them explicitly differentiate action opportunities based on both spatial locations and current state of the world.

3.2.2. Spatial Affordances

Recently, the idea of action opportunities has been extended to describe environments as a combination of spatial affordances, that are used for generating and supporting adaptive agent behaviors. For instance, Kapadia et al. [64] use affordances to select the best action for collision avoidance and to guide an agent's steering decision in a virtual scenario. Luber et al. [83] use affordances to improve tracking and prediction of people destinations. They adopt a place-dependent motion model

that is learned by a spatial Poisson process and whose predictions follow people's space-usage patterns. Similarly, but in a robotic domain, Tipaldi and Arras [140] also use the idea of spatial affordance maps to encode human activities. This model is then exploited to support robot movements, with the goal of covering inhabited environments in a socially acceptable way. In particular, the affordance map is used to represent the presence of people and to avoid crowded areas. Finally, Epstein et al. [35] describe a cognitive architecture that builds a spatial model as a collage of spatial affordances. Such model, combined with simple heuristics, supports effective navigation without a map in near-optimal travel times.

Unfortunately, it must be mentioned that there is not a vast literature regarding the notion of affordances to represent spatial knowledge. No prior work adopts a general framework to model spatial affordances, and typical solutions are task specialized and cannot be generalized to represent the semantics of different activities.

3.3. Approaches to Policy Learning

Machine learning methods have been increasingly used in robotics to deal with uncertain and unstructured environments. In such scenarios, directly learning from data a (sub-)optimal set of parameters to generate robot behaviors is often more robust than hard coding them from prior knowledge. Consequently, due to its practical relevance, policy learning has become a very active area of research. In particular, approaches based on reinforcement learning [89, 143] and Monte Carlo tree search [133], as well as imitation learning [123, 120] have been successfully applied in several contexts and domains.

In this section we review policy learning methods and describe their applications in robotic scenarios. In particular, while analyzing general differences and interrelations between approaches, to correctly collocate our work we focus on their applicability in robotics domains.

3.3.1. Imitation and Reinforcement Learning

Learning based methods for controlling robotic systems [71], or more generally autonomous agents, have become increasingly popular in the recent years [139, 85, 7, 31, 72, 55]. In particular, many approaches have been proposed in the context of learning by imitation – for easily and intuitively instructing robots to execute complex tasks. Asfour et al. [4], for example, instruct a humanoid robot by using continuous Hidden Markov Models (HMMs) that are trained with a set of key points. Chernova and Veloso [24] use a probabilistic policy representation based on Gaussian Mixture Models. In particular, they propose an approach which enables the agent to request demonstrations for specific parts of the state space, achieving increasing autonomy in the execution. Calinon et al. [18], instead, use HMMs for learning a joint distribution of demonstrated positions and velocities, and reproduce the motion through the use of Gaussian Mixture Regression.

In several research projects, traditional imitation learning has been associated with methods for refining the learned policies, as in the case of Nicolescu and Mataric [98]. In particular, additional work has been done to improve learned policies by means of reinforcement learning. Guenter and Billard [50], for example, use

reinforcement learning to relearn goal-oriented tasks with unexpected perturbations. Kormushev et al. [74] encode movements with and extension of Dynamic Movement Primitives [59] initialized from imitation. Kober and Peters [70], instead, use episodic reinforcement learning in order to improve motor primitives learned by imitation for a Ball-in-a-Cup task.

A major effort has been made for incrementally learning behaviors, based on already acquired skills, both through symbolic and numerical [119] approaches. Pardowitz et al. [107, 106], for example, use a hierarchical representation of complex tasks, generated as a sequence of elementary operators (i.e., basic actions, primitives). The method is applied on a robot servant, which has to learn an everyday household task by combining reasoning and learning. A similar approach is used by Ekvall and Kragic [34], who decompose tasks in sub-tasks, which are then used for generalization. Friesen and Rao [40], instead, propose a solution for achieving hierarchical task control by means of an extended Bellman equation. In particular, the authors consider both temporally extended actions (called options) and primitives. Finally, Neumann et al. [97] describe a method for generalizing and learning motor primitive, as well as their selection and sequencing for the execution of complex tasks.

Recently, policy learning methods have been also used in games and video-games to obtain human-level performances. Mnih et al. [89], for example, present a deep agent (deep Q-network), that can use reinforcement learning to generate policies directly from high-dimensional sensory inputs. The authors test their algorithm on classic Atari 2600 games, achieving a level comparable to that of a professional human player across a set of 49 games. Similarly, Silver et al. [133] use deep "value networks" and "policy networks" to respectively evaluate board positions and select moves for the challenging game of Go. These neural networks are trained by a combination of supervised learning from human expert games, reinforcement learning and Monte Carlo tree search. The resulting program showed to be able to beat human Go champions and to achieve a performance beyond any previous expectation.

Dataset Aggregation Differently from previous work, Ross et al. [123, 120] propose a meta-algorithm for imitation learning (DAgger), which learns a stationary deterministic policy that is guaranteed to perform well under its induced distribution of states. Their method is based on data aggregation and strictly relates to no-regret online learning. Similarly, Ross and Bagnell [122] introduce AggreVate and NRPI. The former leverages cost-to-go information – in addition to correct demonstrations – and data aggregation; the latter extends the idea of no-regret learners to approximate policy iteration for reinforcement learning. Based on their work, Chang et al. [22] describe a new learning to search algorithm and provide a local-optimality guarantee.

3.4. Discussion

In this section, we analyze the relations between ideas and approaches previously introduced. In addition, we indicate their connection with respect to the contributions of this thesis.

3.4.1. Relation Between Approaches

Despite adopting different perspectives and alternative goals, few similarities exist between the approaches described in this chapter. For example, both semantic maps and affordance models attempt to represent world semantics for influencing robot behaviors. However, while semantic maps explicitly model information through symbols and labels, affordances implicitly capture under which environmental circumstances an agent can execute an action. Information learned through these representations is hence used for different purposes.

Noticeable, instead, is the relation between affordances and policy learning. In fact, affordance models have been adopted in literature for learning policies [82, 65] and, conversely, object affordances have been retrieved, given an initial policy [146]. For example, in Wang et al. [146] the authors use a simple policy to learn the affordance of an object to be pushed. In Lopes et al. [82] and Katz et al. [65], instead, affordances are exploited to respectively learn action primitives for human's imitation and for autonomous pile manipulation.

We leverage these conceptual and methodological relations between approaches, and we further explore them by (1) interconnecting semantic maps and affordance models, and (2) by further investigating how affordances can be learned and used while generating robot behaviors.

3.4.2. Relation to Thesis Contributions

Semantic Mapping One of the main contributions of this thesis is a principled approach for modeling and representing semantic maps. In fact, as we described in Section 3.1, multiple representations have been proposed in the literature, with alternative structures and few similarities. However, the lack of common formalisms, and consequently the absence of standard evaluation metrics, impede comparative studies and strongly affect the advancements in the field. For this reason, in Chapter 4 we propose a novel basic model to represent semantic maps. By using this model, evaluation metrics can be defined and incremental algorithms can be defined (Chapter 6) to learn semantic knowledge and affect robot behaviors. Further benchmarking solutions are explored, in this field, by describing a procedure for the generation – for the first time – of a semantic mapping dataset (Chapter 4).

A further contribution of this work is represented by the connection that we establish between semantic maps and affordance models. While the former are used to support high-level behaviors, the latter are accessed through semantic maps and implement situated action as structural element of complex plans.

Affordances and Policy Learning An important contribution of this thesis consists in the development of a novel spatio-temporal affordance model. Differently from prior work (Section 3.2), this model leverages spatial semantics (Chapter 5) and accounts for the dynamics of the world. We use spatio-temporal affordances in the context of a representation named "Spatio-Temporal Affordance Map", that explicitly allows the environment to be modeled by means of a semantic map.

Our contribution on affordance models further extends prior work on policy learning (Section 3.3). In fact, by leveraging the literature on data aggregation, in Chapter 7 we investigate how affordances can be used together with Monte Carlo

tree search to generate effective behaviors in the context of situated action. To this end, we extensively make use of simulations. For this reason, in Chapter 8 we also study how learned dynamics models (i.e., transition functions) of the environment can be effectively used for simulation in the context of robot control.

Part II

Models

Chapter 4

A Formalization of Semantic Mapping

Or perhaps no one can understand anyone: each blackbird believes that he has put into his whistle a meaning fundamental for him, but only he understands it; the other gives him a reply that has no connection with what he said; it is a dialogue between the deaf, a conversation without head or tail.

— Italo Calvino

In this chapter, we motivate and present a formal model for semantic mapping [21]. This model builds upon previous definitions [102] and extends existing literature [149, 109, 51, 9, 52, 115] by introducing a *structured approach to semantic mapping*. Our formal model includes a general but flexible framework for map representation, as well as a precise description of the mapping process. Importantly, the absence of a common formalism and, consequently, the lack of proper validation and evaluation methods impose significant limits on the research field. Instead, our formalism prescribes a methodology for obtaining standardized semantic maps, with basic structure constraints and unambiguous error metrics. Of significance for the focus of this thesis, such model provides a principled approach for representing and acquiring semantic knowledge – that can define and influence robot behaviors.

This chapter is organized as follows. First, we introduce our formal model (Section 4.1 and 4.2) of semantic mapping. Then, we discuss its main properties, as well as its advantages for the definition of potential evaluation metrics. Finally, we show an example of use of the proposed model for the generation of a semantic mapping dataset (Section 4.4).

4.1. Knowledge Representation

As introduced in Section 2.1, a semantic map is a representation that enables additional information to be inferred, whenever it is associated with an engine that supports reasoning and behavior generation for an autonomous agent. Multiple

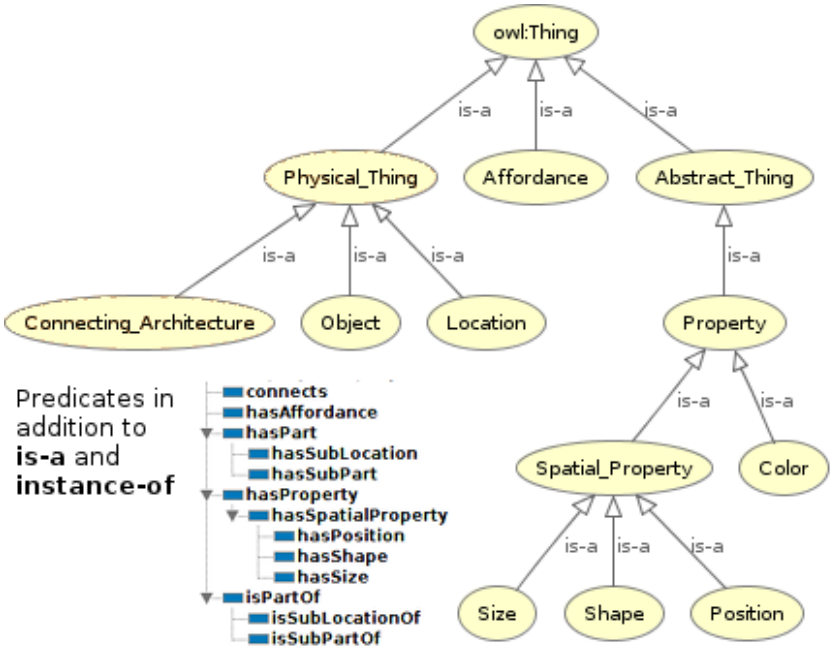


Figure 4.1. Basic concept hierarchy for a semantic map, represented on the Protégé software [95].

approaches have been proposed in the literature (Section 3.1), that cannot be comparatively evaluated as they lack homogeneity and proper evaluation metrics. For this reason, in this section we propose a formalization of a *basic* general structure for the representation of a semantic map. This representation is proposed to play the role of a common interface among all the semantic maps, and can be easily extended or specialized as needed. More specifically, we adopt the word *basic* to emphasize the relation between our formalization and prior work, as our model results from a generalization and intersection effort with respect to the representations adopted in the literature.

The remainder of this section is structured as follows. First we formalize the basic structure of a semantic map (Section 4.1.1). Then, we analyze some of its properties (Section 4.1.2) and we discuss the notion of structural bias (Section 4.1.3).

4.1.1. Formalization: Structure and Specifications

In our general formalization, a semantic map is a representation that is compliant with the basic semantic map model. In order to define this model, however, we first need to introduce the notion of basic concept hierarchies.

Definition 4.1. *A **basic concept hierarchy** is a set of predicates $\mathcal{P}$, included in a knowledge base, that is consistent with the structure proposed in Figure 4.1. These predicates are expressed in a certain representation language, specified by its designer.*

In the concept hierarchy expressed in Figure 4.1, the predicate *is-a* represents the subclass relation, meaning that if *is-a*(`B`, `A`) holds, the class `B` is a subclass of the class

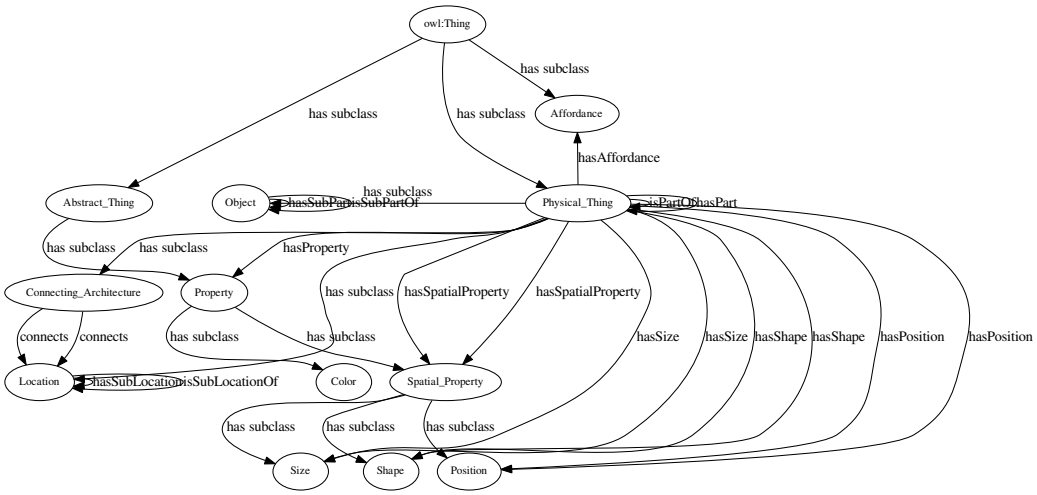


Figure 4.2. Graph of the basic concept hierarchy.

`A` and every instance of `B` is also an instance of `A`. The predicate *instance-of*, instead, represents the membership relation, meaning that if *instance-of*(`a`, `A`) holds, the individual `a` belongs to the class `A`. Additionally, some predicates have a function-like behavior, meaning that they can occur only once for each individual. For example, if dealing with the classes `Person` and `IDNumber`, the predicate *hasId*(`X`, `Y`) occurs only once for each instance of `Person` and `IDNumber`.

Importantly for the purposes of this thesis, the concept hierarchy also presents a predicate *hasAffordance*, that (as illustrated in Figure 4.2) relates the class `Physical_Thing` with the class `Affordance`. This relation will be further detailed in Chapter 5.

Given this definition, first we express a model of basic semantic maps by means of basic concept hierarchies, and then we extend it to the general notion of semantic maps.

Definition 4.2. *A **basic semantic map** is a triple*

$$\mathcal{SM} = (R, \mathcal{M}, \mathcal{P}),$$

where:

- *R is the global reference system, in which all the elements of the map are expressed;*
- *$\mathcal{M}$ is a set of geometrical elements, obtained as raw sensor data. These elements are expressed in the reference frame R and describe spatial information in a mathematical form. $\mathcal{M}_s \subseteq \mathcal{M}$ is the subset of semantically relevant elements available in $\mathcal{M}$;*
- *$\mathcal{P}$ is a basic concept hierarchy. $\mathcal{P}_s \subseteq \mathcal{P}$, with $|\mathcal{P}_s| > 0$, is the set of predicates that provide an abstraction of the elements in $\mathcal{M}_s$.*

Definition 4.3. *A semantic map is a representation that adheres to, and eventually extends, the basic semantic map model.*

In this representation, $\mathcal{P}$ models semantic information that is used to generate robot behaviors. A portion of this knowledge is independent from the domain (i.e., it is general and describes taxonomies useful for inference), while a subset $\mathcal{P}_s$ specifically refers to elements of the environment. The world is geometrically represented through raw sensor data, thus enabling the robot to (1) virtually navigate the environment (through R and $\mathcal{M}$), (2) refine its knowledge by re-evaluating sensor information, and (3) observe relevant features of the world.

To better understand how to apply the proposed model to a general semantic map, let us consider a robot that acts in a mall and interacts with people. In this settings, we can build the desired representation by starting from a basic semantic map and by choosing $\mathcal{M}$ to be a set of 3D points, as acquired from an RGB-D sensor. Equivalently, we can adopt a more complex point cloud that models the map of the mall, but is still composed of 3D points as provided from the sensors. Under these circumstances, $\mathcal{M}_s$ is a subset of $\mathcal{M}$ containing 3D points that represent features of interest, such as shops or objects in the shop. These features can be identified in multiple ways, such as manual labeling or automatic recognition. Then, we can extend the basic concept hierarchy of Definition 4.1 by enriching the set $\mathcal{P}$ according to the following instructions:

- define a class `Person` and add the predicate *is-a*(`Person`, `Physical_Thing`), since a person is an element of interest of the environment;
- specialize the class `Location` for shops and corridors, by defining the classes `Shop`, `Corridor` and adding the predicates *is-a*(`Shop`, `Location`), *is-a*(`Corridor`, `Location`);
- specialize a `Connecting_Architecture` to always use the predicate *connects* for an element of the class `Shop` and one of the class `Corridor`;
- define a class `Advertisement`, add the predicate *is-a*(`Advertisement`, `Abstract_Thing`) and define a new predicate *hasAdvertisement*(`X`, `Y`), where `X` could be an instance of `Shop` and `Y` an instance of `Advertisement`.

Finally, we can add to $\mathcal{P}_s$ a set of predicates that abstract elements in $\mathcal{M}_s$ (e.g., size and position of objects expressed in the knowledge base) and select a reference frame R that corresponds to the global frame of the 3D map used in $\mathcal{M}$, or the one according to which all the points are expressed.

4.1.2. Properties

In this section, we briefly discuss and remark upon some of the properties of the proposed semantic map model.

Remark 4.1. *Definition 4.3 relies on the notion of basic representation. Hence, a general semantic map implements and eventually extends, as best required by the designer, the basic semantic map.*

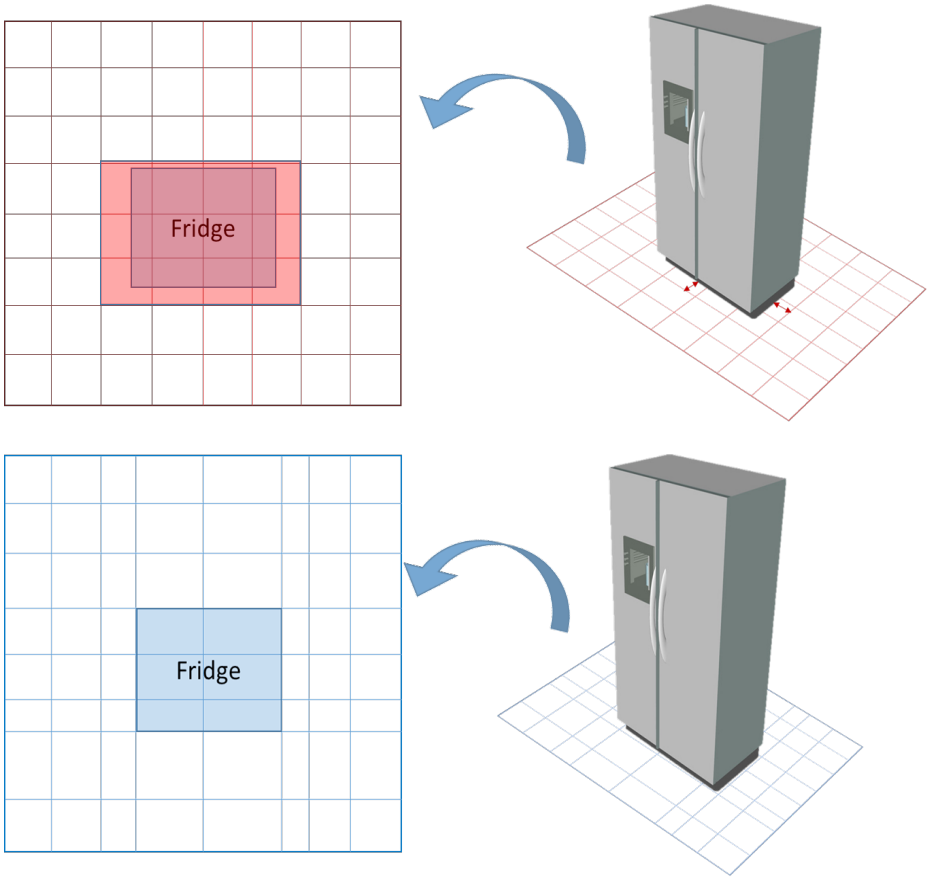


Figure 4.3. Example of structural bias of a semantic map with fixed and adaptive grids.

Remark 4.2. *The use of a unique reference frame R in Definition 4.2 enables the association of the elements of $\mathcal{M}_s$ with those of $\mathcal{P}_s$.*

Remark 4.3. *The use of a set $\mathcal{M}$, composed of geometrical elements obtained as raw sensor data, enables the possibility of retrieving original sensor data, given a pose specified in R.*

Remark 4.4. *Following Remark 4.2 and 4.3, given two semantic maps of the same environment, expressed in the same reference frame R, it is **at least** possible to compare both the predicates and the geometrical elements of the basic parts of the two representations.*

Remark 4.5. *A semantic map stores some parameters that, as described in Chapter 5, describe affordances.*

4.1.3. Representation and Structural Bias

Despite the availability of raw sensor data, the model introduced in Section 4.1.1 is not exempt from approximation errors. These, in fact, can arise from the predicates in $\mathcal{P}_s$ – chosen to symbolically represent $\mathcal{M}_s$ – and, consequently, from the structural bias induced by the abstraction of geometrical elements.

Definition 4.4. *In a semantic map, the **structural bias** is the approximation error that is induced by the predicates $\mathcal{P}_s$ – i.e., the error that depends on the abstraction chosen for symbolically representing spatial information.*

The abstraction represented by $\mathcal{P}_s$, in fact, may consist of (but is not limited to) fixed-size grids, adaptive grids, bounding boxes, etc. Hence, as illustrated in Figure 4.3, the use of different representations introduces different approximation errors on the same spatial information. Intuitively, in a semantic map, the lower the structural bias, the better the quality of the representation.

4.2. The Process of Semantic Mapping

In our model, semantic mapping is the process of gathering information about the environment and generating a representation $\mathcal{SM}$, similar to the one introduced in the previous section. As outlined in Section 3.1, however, an autonomous agent can perform this activity either autonomously, or in collaboration with an user. Still, this process can be formally modeled and uniquely determined in terms of the chosen representation and a sequence of events. We provide such formalization in this section, we extend it to the notion of incremental semantic mapping (Section 4.2.1), and we describe some of its properties.

Definition 4.5. *The process of **semantic mapping** is the implementation of a function*

$$\phi_{SM} : \mathcal{M} \times \mathcal{E} \rightarrow \mathcal{SM}$$

where:

- *$\mathcal{M}$ is a set of geometrical elements, obtained as raw sensor data;*
- *$\mathcal{E}$ is the set of events $\{e_i\}$ that occur in the environment, while the agent is exploring it;*
- *$\mathcal{SM}$ is a semantic map generated according to Definition 4.3.*

Under this perspective, automatic semantic mapping can be seen as a process ϕ_{SM} whose events $\{e_i\}$ originate from an agent's autonomous recognition of objects and locations within the environment. Similarly, when the agent collaborates with an user, events $\{e_i\}$ are generated from different interactions between the robot and the user itself, independently of the interface.

4.2.1. Incremental Semantic Mapping

Independently from the chosen approach, systems proposed in literature are often designed to learn every possible aspect of the environment, through a separate and independent process carried out before task execution. Limits to this perspective, however, immediately arise from the disconnection between robot capabilities and knowledge acquisition. In fact, in this thesis we argue that, in order to be beneficial to the robot, semantic information must be oriented towards autonomous behavior generation. However, all the information in the environment does not necessarily

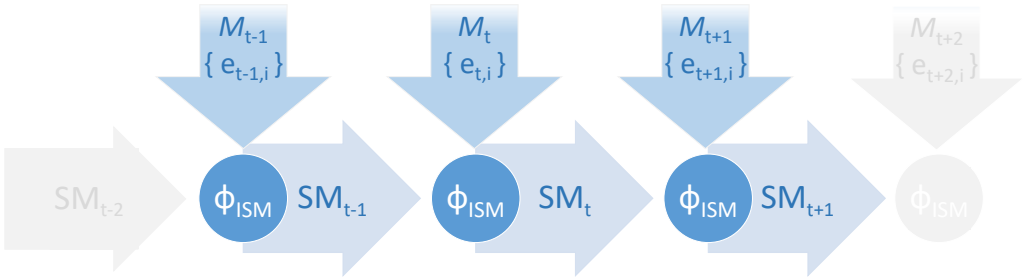


Figure 4.4. Schema of the incremental semantic mapping process.

support robot behavior, and acquiring information before task execution might not be informative for the agent itself. Hence, we consider a robot that does not collect complete information about the world, but incrementally gathers new information, that is integrated within the semantic map.

Definition 4.6. *The process of* ***incremental semantic mapping*** *is the implementation of a function*

$$\phi_{ISM} : \mathcal{M}_t \times \mathcal{E}_t \times \mathcal{SM}_{t-1} \rightarrow \mathcal{SM}_t$$

where:

- $\mathcal{M}_t$ *is a set of geometrical elements, obtained as raw sensor data observed at time* t*;*
- $\mathcal{E}_t$ *is the set of events* $\{e_{t,i}\}$ *that occur at time* t*;*
- $\mathcal{SM}_{t-1}$ *is a semantic map that is generated at time* $t-1$ *according to Definition 4.3.*
- $\mathcal{SM}_t$ *is a semantic map that is generated at time* t *by means of* ϕ_{ISM}*.*

While this process is illustrated in Figure 4.4, under a slightly different perspective the function ϕ_{ISM} can be viewed as a simpler process. This process, at each time t, generates a semantic map $\mathcal{SM}_t$ by means of ϕ_{SM} and then aggregates it as $\mathcal{SM} = \mathcal{SM} \cup \mathcal{SM}_t$. This aggregation process takes care of the consistency of the semantic map – that is, it needs to make sure that no contradictory information is present in the map, at any level.

Remark 4.6. *A semantic map generated through* ϕ_{ISM} *is a dynamic entity that evolves over time, as a consequence of the actions of the agent. Hence, the agent itself partially affects the events* $\{e_{t,i}\}$*.*

Remark 4.7. *Following Remark 4.6, a semantic map generated through* ϕ_{ISM} *is coherent with the paradigm of ecological robotics (see Section 2.2), in the sense that it connects the dynamics of the environment to the collected information about it.*

4.3. Metrics and Benchmarks

Once we are given a basic common structure between two different semantic maps, it is possible hypothesize and define evaluation metrics to be used for comparative evaluations and benchmarking activities. In this section, we briefly investigate this problem by relying on our formal model, and we discuss a solution that is based on two assumptions. First, we assume that a ground truth, given a certain semantic map representation, exists. This implies that one semantic map can be "better" than another, and that the notion of "better" can be quantitatively evaluated. Second, we assume the accessibility of an oracle, that is a perfect correspondence finder between points, lines, and planes that belong to different metric representations of the world, expressed in the same reference frame.

Let us assume to have access to two different semantic maps of the same environment, $\mathcal{SM}_1$ and $\mathcal{SM}_{GT}$, where GT stands for ground truth. Then, given their representations $\mathcal{SM}_1 = (R_{GT}, \mathcal{M}_1, \mathcal{P}_1)$ and $\mathcal{SM}_{GT} = (R_{GT}, \mathcal{M}_{GT}, \mathcal{P}_{GT})$, an evaluation metric can be defined as

$$\delta(\mathcal{SM}_1, \mathcal{SM}_{GT}) = f(|\mathcal{M}_1 \ominus \mathcal{M}_{GT}|, |\mathcal{P}_1 \boxminus \mathcal{P}_{GT}|). \tag{4.1}$$

Importantly, in this definition, the reference frames R_{GT} of $\mathcal{SM}_1$ and $\mathcal{SM}_{GT}$ coincide. This can be easily obtained by applying the transformation offset between the original frame R_1 of $\mathcal{SM}_1$ and R_{GT} of $\mathcal{SM}_{GT}$.

The definition of the operators $\ominus$ and $\boxminus$ determines the metric itself, that finally results in a linear combination of the geometric and predicate errors. In particular, the operator $\ominus$ can be implemented as a distance d between geometrical elements in $\mathcal{SM}_1$ and $\mathcal{SM}_{GT}$ (see Table 4.1). This, however, is realizable only by means of the oracle. The operator $\boxminus$, instead, can be realized to return two sets of predicates, Δ and Γ, such that:

$$\{\mathcal{P}_1 \setminus \Gamma\} \cup \Delta \models \mathcal{P}_{GT}. \tag{4.2}$$

Here, the lower the cardinality of Δ and Γ, the better the semantic representation. Still, a similar metric does not consider the subset $\mathcal{P}_s$ as containing references to spatial information (which can be measured through metric criteria). A solution to this problem can be found in the redefinition of the $\boxminus$ operator to return two sets of predicates, Δ and Γ, as well as a distance d such that:

$$\{(\mathcal{P}_1 \setminus \mathcal{P}_{1_s}) \setminus \Gamma\} \cup \Delta \models \{\mathcal{P}_{GT} \setminus \mathcal{P}_{GT_s}\}, \quad d(\mathcal{P}_{1_s}, \mathcal{P}_{GT_s}). \tag{4.3}$$

Table 4.1. Example definition of the $\ominus$ operator for implementing an error metric for semantic maps. The index i indicates the i-th corresponding geometric element in $\mathcal{M}_1$ and $\mathcal{M}_2$, while p, l and π represent respectively a point, a line and a plane.

		$\mathcal{M}_{GT}$		
		Points	Lines	Planes
$\mathcal{M}_1$	Points	$\sum_i d(p_i^1, p_i^{GT})$	$\sum_i d(p_i^1, l_i^{GT})$	$\sum_i d(p_i^1, \pi_i^{GT})$
	Lines	$\sum_i d(l_i^1, p_i^{GT})$	$\sum_i d(l_i^1, l_i^{GT})$	$\sum_i d(l_i^1, \pi_i^{GT})$
	Planes	$\sum_i d(\pi_i^1, p_i^{GT})$	$\sum_i d(\pi_i^1, l_i^{GT})$	$\sum_i d(\pi_i^1, \pi_i^{GT})$

For example, let us consider a semantic map $\mathcal{SM}_{GT}$ (that is a ground truth), containing a table and a chair, that are correctly positioned in the representation $\mathcal{SM}_{GT} = (R_{GT}, \mathcal{M}_{GT}, \mathcal{P}_{GT})$. Given a different semantic map $\mathcal{SM}_1$, and in the case in which the table is missing in the set $\mathcal{P}_1$ of $\mathcal{SM}_1$, according to our metric in Eq. 4.2 we obtain $|\Delta| = 1$. Indeed, in this case, a robot endowed with the semantic map $\mathcal{SM}_1$ cannot execute the command "go to the table". Conversely, if the table-related information belongs to $\mathcal{P}_s$, then we have $|\Delta| = 0$ and the robot can use $\mathcal{SM}_1$ to execute the requested task. Similarly, if the object is not well positioned in $\mathcal{M}_1$ any distance from Table 4.1 would be much bigger than zero, and the robot would execute the command by reaching an incorrect location. Additional metrics could be defined on different criteria like the processing time, the distance traveled by the robot, the number of sensor readings processed, etc.

While the metrics (and the model) presented in this thesis do not explicitly address the problem of synonyms in the set $\mathcal{P}$, these should not be penalized as long as each predicate is consistently modeled. Although considering different synonyms among multiple semantic maps increases the complexity of the process, the resulting evaluation is then unbiased on the chosen terminology. Alternatively, a vocabulary should be agreed or generated before the evaluation process itself.

As for synonyms, the model does not explicitly account for uncertainty in the representation. Still, uncertainty can be modeled as part of property predicates in the knowledge base $\mathcal{P}$, for each instantiated element. In this way, every evaluated predicate can be weighted by the corresponding probability measure to contribute in determining a final error metric.

4.4. Using the Model: Semantic Mapping Dataset

Once we are given a formal model for semantic mapping, and a principled approach for knowledge acquisition (see also Chapter 6), our intent consists in (1) carrying out this activity in an environment, and (2) comparing the obtained semantic map against a ground truth, according to some metrics. Unfortunately, while some Robotics Innovation Facilities exist[1] for this purpose, it is still not easy to find locations and environments to perform comparative evaluations. This is due to logistic, physical and economic constraints. For these reasons, in this section we present a first application of our model, and we discuss the development of a dataset of semantic maps, that accounts for the schema introduced in this chapter.

This section is structured as follows. First, we describe the motivations for using our model to generate a semantic mapping dataset (Section 4.4.1, and how this can be accomplished (Section 4.4.2). Finally, we show an example of dataset acquired through the process described in this section 4.4.3.

4.4.1. Model and Dataset

In this section we describe the motivations for the generation of a dataset for semantic maps, given a representation $\mathcal{SM}$, and how this activity can be carried out in a methodological way.

[1] `http://echord.eu/rif/`

Let us consider the three elements of the representation $\mathcal{SM} = (R, \mathcal{M}, \mathcal{P})$. From previous sections in this chapter, we already know that $\mathcal{M}$ describes a set of geometrical elements expressed in R, generated as a collection of raw sensor data. This property of the representation is, in practice, a strong motivation for its use in the construction of a dataset. In fact, since we need to account for the impossibility of moving a robot to a certain location, we are interested in the opportunity of simulating both its navigation and sensor acquisition directly through the dataset. The availability of raw sensor data within the representation, then, makes this activity simple to implement. This can be done by defining a projection function that transforms the elements of $\mathcal{M}$ into the associated sensor domain. For example, in the case of a RGB-D camera the geometrical elements can be projected in a depth and RGB image, while in the case of a laser they can projected into a vector of range values. Nevertheless, from the definition of semantic map, we also know that the predicates $\mathcal{P}$ of its representation implement, at least, a basic concept hierarchy that is common to all semantic maps. Although this is not intended to represent full information about the environment, a basic representation enables a direct comparison among methods, that is one of the goals of having a ground truth.

Given these motivations, in the next section we proceed in describing how a dataset can be constructed according to the structure of our representation.

4.4.2. Dataset Generation

In this section, we illustrate our[2] method for the generation of a ground truth, in which the set $\mathcal{M}$ consists of a 3D point cloud, $\mathcal{P}$ implements the basic concept hierarchy and $\mathcal{P}_s$ contains abstractions of bounding boxes.

To collect geometric information, we consider a set of low-cost[3] and common sensors (i.e., RGB-D cameras). Then, to generate the point cloud through our software[4][5], we leverage a semi-automated process that, on some occasions, requires the intervention of a human expert.

As shown in Figure 4.5, this process is composed of several steps, which can be articulated in terms of metric and semantic phases. First, we acquire data in order to generate a 3D map and we perform a preliminary manual annotation of the objects inside the environment. Then, by associating semantic information and volumes in the 3D map, in the form of bounding boxes, we obtain the desired semantic map. These steps are detailed in the remainder of this section.

Data Acquisition As previously introduced, the activity of acquiring data can be separated in two different phases, one related to the 3D map, the other to the semantic annotations of the elements of interest. While manually collecting semantic annotations is relatively easy, although tedious, 3D data acquisition results more challenging due to the limitations of low-cost sensors.

[2] Although we are proposing a practical methodology (and an example) for the construction of a dataset, alternative approaches may be used if compliant with the model of semantic map.

[3] Note that building a 3D map with this kind of sensors, leads to multiple open issues.

[4] `http://www.dis.uniroma1.it/~labrococo/nicp`

[5] `http://www.dis.uniroma1.it/~labrococo/sem_map_dataset`

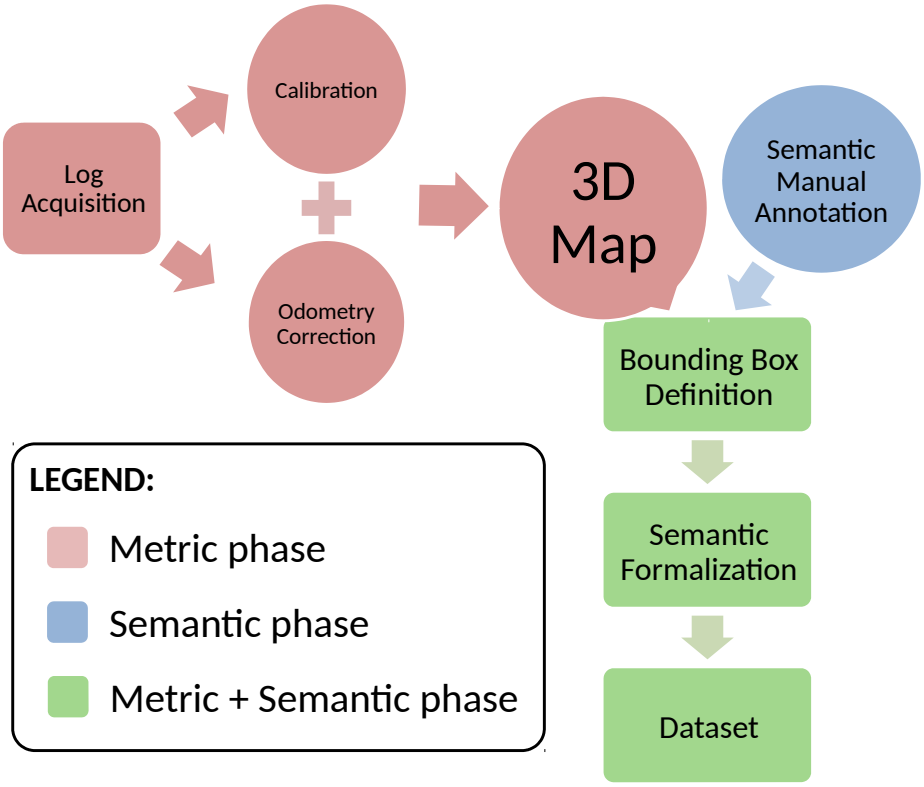


Figure 4.5. Steps for the construction of a semantic mapping dataset.

The generation of a 3D map, in fact, requires the acquisition of a log containing data from the sensors of a robot, while moving around the environment. In particular, the log should contain the robot odometry (or laser data) and the camera stream (both for depth and RGB information). Additionally, if using our software for mapping the environment, while taking the log one should pay attention to steer the robot in such a way that at least one camera does not see only a flat surface. Indeed, structures like a floor, a wall or two parallel planes do not help the mapping system, due to their poor geometrical information.

Sensor Calibration The calibration of a sensor consists in correctly computing its internal parameters, as well as its pose with respect to the robot reference frame. Extracting the right internal parameters improves the data generated by the sensor and reduces its intrinsic error. For example, in the case of a depth camera, this corresponds to determining its camera matrix and distortion parameters. Computing the correct pose of a sensor allows to accurately express data measurements with respect to a different reference frame.

In order to perform sensor calibration, and supposing n RGB-D cameras on the robot, $n + 2$ logs[6] are required. In particular, choosing one of the cameras as a *reference*, we have:

[6] A log is obtained by acquiring and recording the required sensor data.

1. *n intrinsic calibration logs*, containing the stream of the *i*-th RGB-D sensor, for the calibration of the internal parameters of its depth camera (refer to Cicco et al. [25] for more details on how to acquire data);

2. 1 *sensor-base calibration log*, containing the robot odometry (or laser data) and the camera stream, for calculating the pose between the robot and reference RGB-D sensor (the robot should slowly translate and rotate while the reference sensor sees at least 3 planes, each of them being non parallel with all the others);

3. 1 *sensor-sensor calibration log* (at least), containing the stream of the n cameras, for computing the pose of $n-1$ RGB-D sensors with respect to the reference one (all the cameras should see, at least once, the same part of the environment while *always* respecting the condition described in point 2).

Common RGB-D cameras are affected by a substantial distortion in the depth channel. Not considering this distortion leads to systematic drifts in the estimate of the robot pose while mapping. This calibration is performed by following the procedure explained by Di Cicco et al. [25] on the intrinsic calibration logs. At the end of this procedure, it is possible to reduce the intrinsic error which normally affects the sensors data (i.e., walls that should be flat, look curved on the edges).

Another goal of the calibration procedure is to find (1) the pose of one of the cameras (*reference*) with respect to the robot frame, and (2) the relative offsets (translation and rotation) between all the other cameras and the *reference*. The software we developed provides two different tools to compute these offsets. The first one computes the transform $\mathbf{T}^*$ between the robot frame and the reference depth camera. By using the sensor-base calibration log we estimate the motion of the camera in a small region. Taking as reference the odometry of the robot, this tool casts a least square problem that minimizes a cost function which depends on the sensor transform $\mathbf{T}$ and returns $\mathbf{T}^*$. The second tool, instead, allows the computation of the offset between pairs of depth cameras. The main idea is to use the sensor-sensor calibration log to generate, for each camera, an independent point cloud. In this way, each sensor produces a cloud starting from its own reference frame. Once this is done, our registration algorithm can be run between pairs of point clouds. The output of the alignment determines the relative translation and rotation between the origins of the point clouds and thus between the sensors. At the end of the calibration we are able to construct a tree of sensor pose transformations (Figure 4.6). From this tree, it is possible to compute the transformation between any two nodes, by a simple offset concatenation.

Mapping and Data Processing Once all the data is acquired, the 3D map can be built. To this end, the point clouds recorded in the log are aligned generating a set of *local maps*. A local map (Figure 4.7) is a point cloud constructed by aligning and integrating, in the same reference frame R, a sequence of sensor data acquired while the robot moves in the environment. This is obtained through the use of a point cloud registration algorithm based on the work by Serafin and Grisetti [130, 131]. A new local map is started whenever one of the two following statements holds:

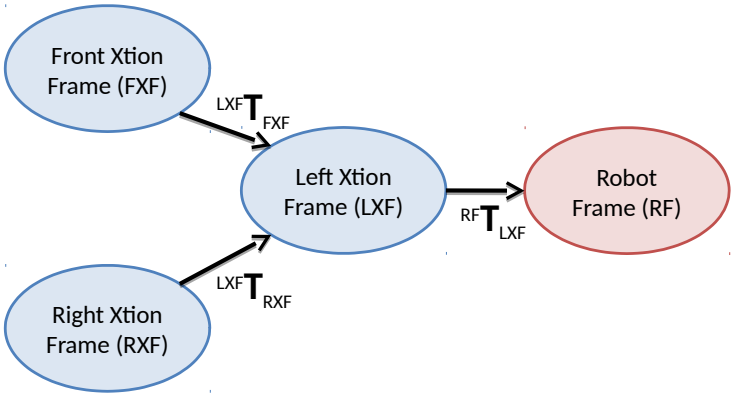


Figure 4.6. Sensor transformation tree generated at the end of the calibration procedure for a robot equipped with 3 depth cameras.

- the estimate of the robot (or equivalently the camera) movement is greater than a certain amount. This allows to limit the growth of the local map in terms of dimension;
- the point cloud registration algorithm detects that the last alignment is not good (with possibility of inconsistency). This is necessary in order to avoid to introduce errors inside the local map.

The local map generator uses the robot odometry as an initial guess for the point cloud alignment. However, a good odometry estimation is not always available. In this case (but this is useful in general), if the robot comes with a 2D laser, it is possible to "correct" the odometry, and use as an initial guess the transformation provided by the *scan matcher*[7] developed as part of our software.

The 3D map is represented as a pose graph [49], where each local map is connected to the previous and following one by means of a transformation. In more detail, nodes of the pose graph represent local maps, with their position and orientation in a global frame R. Edges are relative transforms between local maps. The benefits of this metric representation are that it allows to add/remove anytime information and update an existing map. In this way, inconsistencies in the map can be manually fixed.

Combining Geometric and Semantic Data Once both the 3D map and the semantic annotations are available, it is possible to combine them by means of a geometric abstraction like a volume in the map. For our dataset, we define such a volume to be a bounding box containing all the geometric elements to which we want to attach the same semantic information. After all the bounding boxes are assigned, we formalize the predicates $\mathcal{P}$ (compliant with the basic conceptual hierarchy) in OWL-DL, using Protégé[8] [95]. Bounding boxes, in particular, belong to the subset $\mathcal{P}_s$ and are formalized by means of classes like `Size`, `Position` and `Shape`.

[7] `https://github.com/webrot9/thin_scan_matcher`
[8] `http://protege.stanford.edu/`

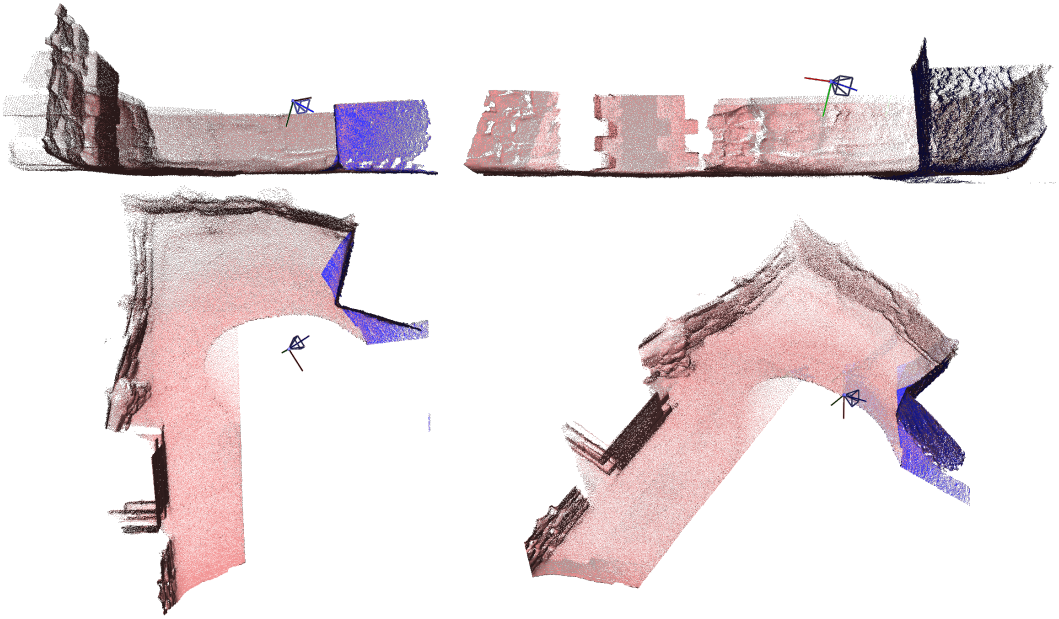


Figure 4.7. Example of a local map and its internal trajectory.

4.4.3. Semantic Mapping Dataset and Scenarios

In this section, we briefly discuss the contexts in which these tools and ideas have been developed, and we additionally showcase some portions of the dataset that we collected.

ECHORD++ Robot Innovation Facility of Peccioli We performed the procedure described so far on a set of data specifically acquired during the RoCKIn Camp[9] held in the ECHORD++ Robotic Innovation Facility of Peccioli[10], in Italy. This is a domestic environment with several rooms and everyday objects built to foster benchmarking of robotic applications, test their robustness, and support standardization efforts. While Figure 4.8 and 4.9 illustrate details of the labeled 3D map of the environment, the whole dataset is hosted online (`http://www.dis.uniroma1.it/~labrococo/sem_map_dataset`) and contains ground truth data that is compliant with the structure of our model of semantic map. Namely, a 3D point cloud with an associated reference frame and the corresponding OWL-DL ontology compose the first example of a dataset for semantic maps. The ontology contains multiple labeled objects (among which fixtures, devices and furniture), 30 classes (Figure 4.10), 16 object properties and, all in all, $\sim$ 200 instances.

Benchmarking competitions The idea of developing a dataset for evaluating semantic maps was generated from the participation to the RoCKIn robot challenge, in the context of the FP7-ICT-601012 EU project. In this scenario, multiple teams were required to develop a semantic mapping algorithm for the participation to the challenges of the competition. While the evaluation of teams' scores depended on

[9] `http://rockinrobotchallenge.eu/`
[10] `http://echord.eu/the-peccioli-rif`

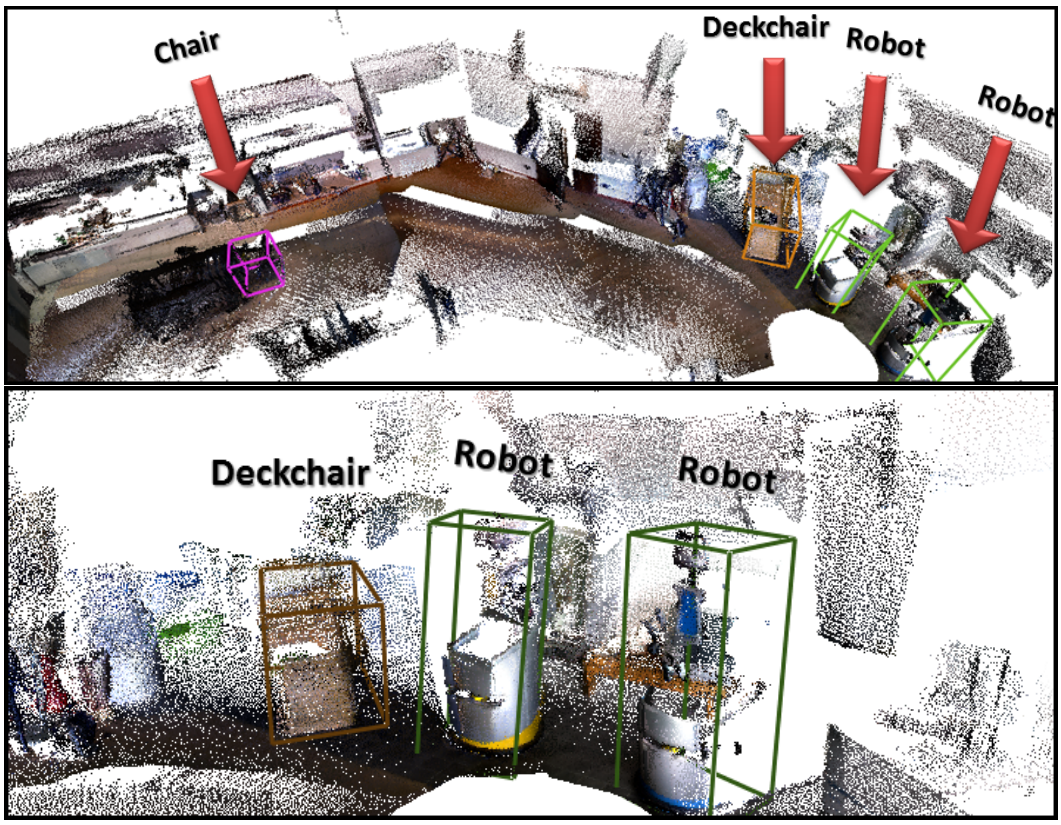


Figure 4.8. Double view of the example dataset acquired in the Robot Innovation Facility of Peccioli, in Italy. Part of the sitting room and the kitchen are shown, together with some bounding boxes identifying a chair, a deckchair and two robots.

the robot capabilities in task execution, the observation of this activity led to the formalization of the semantic mapping model.

4.5. Contributions

In this chapter, we motivated and presented a formal model for semantic mapping, that we also leverage in Chapter 6. In particular, we designed a formalization for representing semantic maps (Definition 4.3 and 4.2), which includes both spatial and semantic knowledge (Definition 4.1), and we analyzed some of its properties together with the issue of the structural bias (Definition 4.4). Additionally, we presented a formal description of the semantic mapping process, and made it incremental (Definition 4.6) for the purposes of robot's behavior generation. On top of this, we discussed some hypotheses for metrics and evaluation criteria, based on the idea that a ground truth for semantic maps exists or can be modeled. Finally, we adopted our model for semantic mapping to build a dataset from real sensor data. This dataset allows to simulate robot navigation inside the environment, thus breaking down logistic, physical, and economic barriers for a fair comparison between different semantic mapping methods.

Our main contributions, in this chapter, are represented by (1) a novel formal

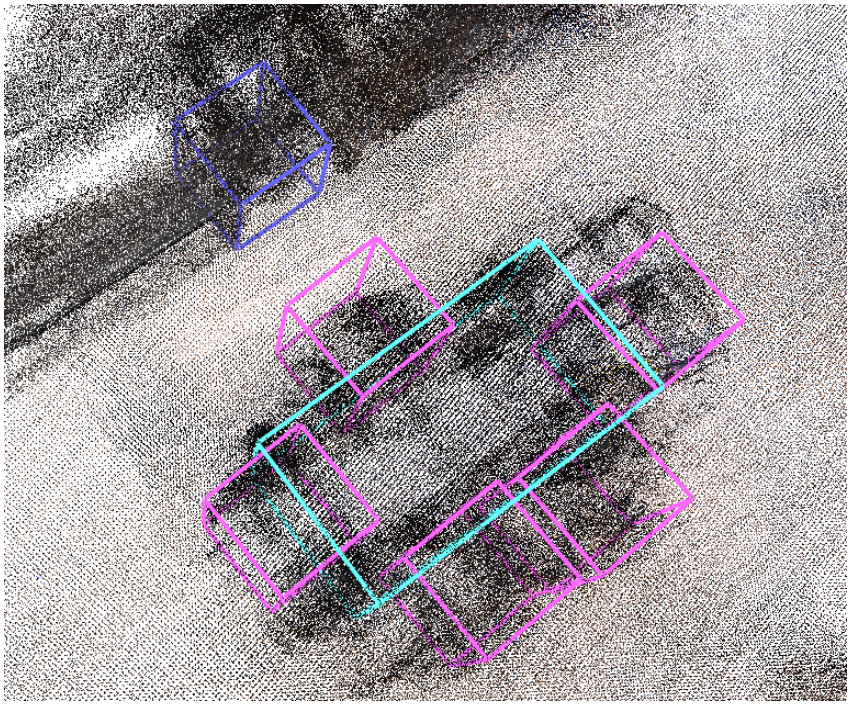


Figure 4.9. Detail of the example dataset acquired in the RIF of Peccioli. The image shows a table and chairs with their associated bounding boxes. RGB information is intentionally omitted and resolution is reduced for a better visualization of the bounding boxes.

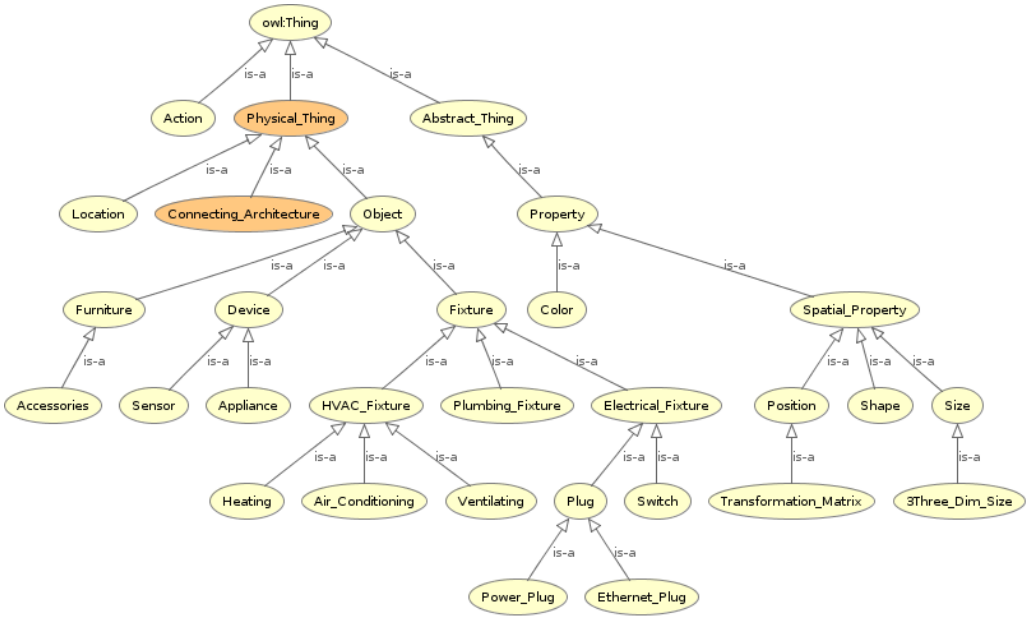


Figure 4.10. Graph representing the classes implemented in the dataset acquired in the RIF of Peccioli. The graph is visualized from Protégé.

representation that explicitly models the environment through the notion of semantic maps; (2) the formalization of a process that adopts this representation and allows its update; (3) the first (to the best of our knowledge) description of procedures and tools for evaluating and benchmarking semantic maps. The work in this chapter further extends our previous publication [21] by discussing in detail properties and characteristics of the model, and by connecting it to the notion of affordances.

Chapter 5

A Model for Spatio-Temporal Affordance Maps

It's the action, not the fruit of the action, that's important. You have to do the right thing. It may not be in your power, may not be in your time, that there'll be any fruit. But that doesn't mean you stop doing the right thing. You may never know what results come from your action. But if you do nothing, there will be no result.

— Mahatma Gandhi

In this chapter, we formalize a model of Spatio-Temporal Affordance Maps (STAM). This model builds on ecological theories [46], as well as on the multiple application of affordances in robotics [94, 28, 142, 91, 67, 35]. In particular, we propose an extension of previous approaches that directly establishes a connection between the environment, its spatial semantics, and their evolution. Differently from previous work, our model can combine multiple affordances to supports manifold tasks. Importantly for the focus of this thesis, Spatio-Temporal Affordance Maps encode the semantics of each action and can be directly used to affect the decision making process of an autonomous agent.

This chapter is organized as follows. First, we introduce the notion of Spatio-Temporal Affordances (Section 5.1.1), and we discuss their modeling, main properties and relation to other approaches. Then, we use this definition to characterize Spatio-Temporal Affordance Maps (Section 5.2) and to provide the reader with an explanatory example (Section 5.3).

5.1. Spatio-Temporal Affordances

As detailed in Section 2.2, affordances have been originally introduced by Gibson [46] as action opportunities that objects offer, and further explored by Chemero [23] in a more recent work. This notion has been accordingly adopted in robotics (Section 3.2) to represent objects and their related actions. Here, we extend the spatial affordance

theory, where the considered "object" is the environment itself, by using the idea of spatial semantics as directly connected to the environment, its dynamics and its operational functionalities. Let us consider, for example, a museum endowed with multiple rotating cameras and a surveillance task. At every moment, areas of the museum that are not entirely covered by the sensors present a different "risk semantics". Since the cameras rotate, and they belong to the environment, our risk changes over time as a function of the environment itself. In addition, there might areas of the museum that are hidden and present a (high) static risk. To model this type of spatial semantics, in this section we introduce the concept of Spatio-Temporal Affordances.

As further discussed in this chapter, spatio-temporal affordances can be directly connected with semantic maps. The goal, in this sense, is to adopt a representation that supports deliberation (semantic maps) to also provide a bridge between symbolic and procedural knowledge. At the level of situated action, reflection occurs implicitly both through spatio-temporal affordances and the intervention of the semantic maps. The former, in fact, express the desirability of an action given its long term value in a certain state (i.e., they enable planning), while the latter provides semantically relevant state information that modify the desirability of an action as a function of the environment.

The remainder of this section is organized as follows. First, we define a Spatio-Temporal Affordance (Section 5.1.1) and, then, we describe the idea of "affordance signature", and its relation to learning or optimization procedures (Section 5.1.2). Finally, we briefly discuss how prior knowledge can be enforced in spatio-temporal affordances (Section 5.1.3), some properties of the proposed model (Section 5.1.4), as well as its relation with other approaches (Section 5.1.5).

5.1.1. Definition of Spatio-Temporal Affordance

A Spatio-Temporal Affordance (STA) is a function that defines areas of the operational environment that afford an action, given a particular state of the world. We recall that, according to Definition 2.11, an affordance is a value function that represents the desirability of an action over a full environment. More formally:

Definition 5.1. *A **spatio-temporal affordance** is a function*

$$f_{E,\mathcal{T}} : \mathcal{S} \times \Theta \rightarrow \mathrm{A}_E \tag{5.1}$$

where:

- E *is the environment;*
- $\mathcal{T}$ *is the set of tasks that are defined for the environment;*
- $\mathcal{S}$ *is the state space of the domain;*
- Θ *is a space of parameters that characterize the affordance function;*
- A_E *is a map over the space of* E *that evaluates the affordance (i.e., the value) of* $\{\tau_j\} \in \mathcal{T}$ *in state* $s_E(t) \in \mathcal{S}$ *at time* t*.*

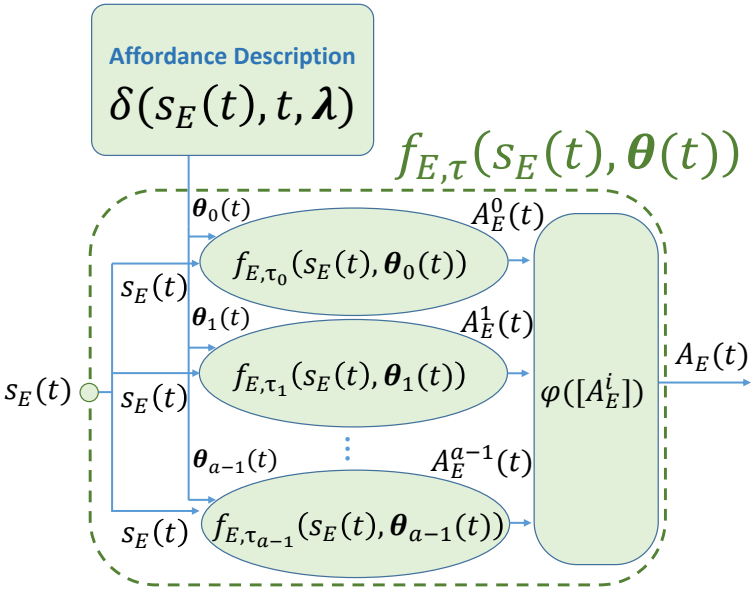


Figure 5.1. Decomposition and detailed overview of a spatio-temporal affordance.

The function $f_{E,\mathcal{T}}(s_E(t), \boldsymbol{\theta}(t))$ characterizes spatial semantics by evaluating areas of E with respect to tasks $\mathcal{T}$, and by generating, at each time t, the spatial distribution of affordances as a map A_E. To do so, the spatio-temporal affordance takes as input the state of the environment $s_E(t)$ (or, for simplicity of notation, s_t) and a vector of parameters $\boldsymbol{\theta}(t) \in \Theta$ (or $\boldsymbol{\theta}_t$) that, as explained later in Section 5.1.2, change over time and represent the affordance signature. Intuitively, these parameters enable our model to consider the state of the environment and to model affordances accordingly, as in our example in the museum. Back to that scenario, in fact, the museum represents the environment E, while the surveillance task is the only element of $\mathcal{T}$. Given our model, a state s_t represented by the rotation angle of each camera, and some parameters $\boldsymbol{\theta}_t$, the STA function emits a map that represents the risk semantics of the environment - i.e., the value of focusing the attention of the cameras on each area of E. This value map can be directly adopted by the agent to support decision making and action planning.

In the case of multiple tasks, a spatio-temporal affordance can be decomposed, as in Figure 5.1, in multiple functions $f_{E,\tau_j}(s_t, \boldsymbol{\theta}_{t,j})$, with $j = 0 \dots \alpha - 1$ and where α is the number of affordances. In particular, each function f_{E,τ_j} models the spatial distribution A_E^j representing the affordance map of a task $\tau_j \in \mathcal{T}$ defined in E. These spatial distributions are then evaluated by an additional function ϕ, that takes as input all the A_E^j and outputs the affordance map A_E of $\mathcal{T}$. In our example, let us assume our autonomous agents – i.e., the cameras in the museum – are assigned both a surveillance and a tracking task. The former, as before, requires the cameras to observe the environment, without leaving portions of it uncovered for a long time. The latter, instead, consists of recording and tracking the visit of a prime minister in

the museum. By using our model, the spatial distributions A_E^j of affording each task are independently encoded by two STA functions. Then, to perform the assigned tasks, these are combined by means of ϕ in a unique map A_E, that enables each camera to choose its best orientation depending on the position of the prime minister and the angles of the other agents. Hence, in this example, ϕ simply consists in the combination of the two affordance distributions.

5.1.2. Affordance Signature and Function Optimization

As introduced in the previous section, the affordance signature is represented by a vector of parameters $\boldsymbol{\theta}_t \in \Theta$, that are dynamically generated at each time instant t. More formally:

Definition 5.2. *The **affordance signature** is the output $\boldsymbol{\theta}_t \in \Theta$ of an affordance description function*

$$\delta_E : \mathcal{S} \times N \times \Lambda \to \Theta \tag{5.2}$$

where:

- $\mathcal{S}$ *is the state space of the domain;*
- N *is the set of time-steps $t = 0 \dots T$;*
- $\boldsymbol{\lambda} \in \Lambda$ *is a vector of parameters that we call affordance descriptors;*
- Θ *is the space of affordance signatures.*

The function $\delta_E(s_t, t, \boldsymbol{\lambda})$ is a mapping from states to affordance signatures, that associates the state of the environment to the evolution of $\boldsymbol{\theta_t}$ at a given time-step, by means of a parametrization $\boldsymbol{\lambda}$. In this way, the signature $\boldsymbol{\theta_t}$ evolves together with the states of the world and the time, to properly contribute in generating affordance distributions. This is beneficial in complex domains to facilitate affordance learning (as a two-step prediction) and address the difficulty of estimating a unique signature for highly dynamic environments. On the other hand, δ_E generates $\boldsymbol{\theta_t}$ as a function of some parameters $\boldsymbol{\lambda}$, that can be designed by an expert (and eventually be optimized) or directly learned by the agent.

The composition of $f_{E,\mathcal{T}}$ and δ_E enables a rich and flexible representation, but also increases the difficulty of designing or learning the values of $\boldsymbol{\lambda}$. In fact, designing appropriate affordance descriptors requires an accurate understanding of both the functions δ_E and $f_{E,\mathcal{T}}$. To tackle these difficulties, we propose a two-step learning procedure that first learns a sequence of $\boldsymbol{\theta}_t$, for $t = 0 \dots T$, and then generates the parameters $\boldsymbol{\lambda}$ using this sequence. In particular, the signatures $\boldsymbol{\theta}_t$ can be learned by means of expert demonstrations or a general reinforcement learning paradigm, with the goal of collecting, for each t, multiple observations of state-task pairs in a dataset $\mathcal{D}_t = \{(s_t, \tau_j)\}$. At time t, by using $\mathcal{D}_t$ and for all tasks $\tau_j \in \mathcal{T}$, the affordance signature $\boldsymbol{\theta}_{t,j}$ is chosen to maximize the summed likelihoods to afford τ_j, in all the states where τ_j is observed in $\mathcal{D}_t$. More formally:

$$\boldsymbol{\theta}_{t,j} = \arg\max_{\boldsymbol{\theta}_{t,j}} \sum_{\{s_t : (s_t, \tau_j) \in \mathcal{D}_t\}} \log f_{E,\tau_j}(s_t, \boldsymbol{\theta}_{t,j}), \tag{5.3}$$

where $j = 1 \dots |\mathcal{T}|$. This optimization consists of finding our parameters $\boldsymbol{\theta}_{t,j}$ by maximizing the likelihood of the portion of the dataset labeled with the considered task τ_j. Intuitively, the higher the likelihood to afford the tasks in $\mathcal{D}_t$, the better the function signature encodes the spatial semantics of τ_j in E at time t. Once a dataset of $\{\boldsymbol{\theta}_t\}$ is collected, it can be used for learning the parameters $\boldsymbol{\lambda}$ of a regression function that maps each state s_t to the corresponding $\boldsymbol{\theta}_t$.

5.1.3. Prior Knowledge and Spatio-Temporal Affordances

We already argued that the flexibility of spatio-temporal affordances comes at the cost of loosing the possibility to manually design appropriate descriptors, and consequently affordance signatures. In fact, this is a difficult, tedious, and unpractical task. For this reason, in this section we briefly discuss whether and how prior knowledge can be enforced in an STA. In particular, although this process is possible, it is not straightforward and unfortunately requires the user to collect or generate a dataset of appropriate state-task pairs.

In our experience with spatio-temporal affordances (Chapter 7), in fact, we have been able to enforce prior information in our models by initializing the descriptors $\boldsymbol{\lambda}$ by means of a dataset $\mathcal{D}_{I_1}$. Such a dataset is supposed to represent the desired action semantics through few, relevant[1] state-task pairs, that can be manually collected (e.g., by means of demonstrations) or artificially generated. Then, the dataset $\mathcal{D}_{I_1}$ can be used to learn a signature $\boldsymbol{\theta}_I$ and, in turn, to initialize $\boldsymbol{\lambda}$. This last step, in fact, can be realized by generating an additional dataset $\mathcal{D}_{I_2}$ composed of all the states previously collected in $\mathcal{D}_{I_1}$, but labeled with the signature $\boldsymbol{\theta}_I$.

More simply, prior knowledge can be enforced at learning time, by aggregating the dataset $\mathcal{D}_{I_1}$ to each $\mathcal{D}_t$, such that $\mathcal{D}_t = \mathcal{D}_t \cup \mathcal{D}_{I_1}$ is used for generating the parameters $\boldsymbol{\theta}_t$.

5.1.4. Properties of Spatio-Temporal Affordances

In this section, we briefly discuss and remark some of the properties of the proposed STA model.

Remark 5.1. *Spatio-temporal maps encode spatial semantics in the probability map* A_E*, depending on the considered tasks* $\mathcal{T}$*.*

Remark 5.2. *In dynamic environments, the affordance signature* $\boldsymbol{\theta}_t$ *is a time-variant system that explicitly depends upon time.*

Remark 5.3. *The mapping* δ_E *enables the probability map* A_E *to evolve according to the dynamics of the system – i.e. the environment and its states.*

Remark 5.4. *Eq. 5.3 is de facto a procedure for finding a maximum likelihood estimate of the parameters* $\boldsymbol{\theta}_{t,j}$*. By assuming unobserved data-points, and depending on the chosen implementation of the function* f_{E,τ_j}*, the desired parameters can be obtained by using an Expectation-Maximization procedure.*

[1] Although the selection of relevant state-task pairs might be a difficult activity, since we are attempting to enforce prior knowledge we assume the availability of a domain expert.

Remark 5.5. *In static environments the affordance distribution does not change over time and, consequently, the affordance signature is time-invariant:* $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t+\delta}$.

Remark 5.6. *Following Remark 5.5, when the affordance signature is time-invariant, the learning process can be reduced to a single step – in which a single vector of parameters* $\boldsymbol{\theta}$ *is estimated over a unique dataset* $\mathcal{D}$ *as*

$$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta}_j} \sum_{\{s_t:(s_t,\tau_j)\in\mathcal{D}\}} \log f_{E,\tau_j}(s_t, \boldsymbol{\theta}_j). \tag{5.4}$$

5.1.5. Relation to Other Approaches

The model of spatio-temporal affordances can be related to methods that leverage the theory of Inverse Reinforcement Learning. Both approaches, in fact, are ultimately adopted to optimize the behavior of an agent, with respect to some semantics that are indirectly captured from the environment. At a closer look, however, some differences emerge that make the two approaches substantially different. Methods based on inverse reinforcement learning, in fact, typically rely on the notion of reward, that is obtained as a linear or non-linear combination of some features [152, 151] provided by the world. The goal of these approaches, then, is to optimize the expectation of such features, given some state-action trajectories demonstrated by a human, or a generic agent. Conversely, spatio-temporal affordances directly optimize the value (the desirability) of a certain action, given a state of the world belonging to a trajectory. This can be done not only by using a set of demonstrations, but also through autonomous agent exploration (as shown in Chapter 7). Hence, the goal of STA concerns more the evaluation of actions' "quality", according to their effects (see also Section 5.2.2), rather than the notion of reward. Therefore, under this perspective, spatio-temporal affordances are more similar to the notion of "forecast values" [127], although their use is different and their representation is non-hierarchical.

5.2. Spatio-Temporal Affordance Maps

Once a proper parametrization is chosen for the STA and the affordance description, these functions must be instantiated to a representation of the environment over which the affordance distribution can be generated. In this section we introduce the Spatio-Temporal Affordance Map, that is a representation in which the descriptors $\boldsymbol{\lambda}$ (or the signature $\boldsymbol{\theta}$) can be learned, updated and used by an autonomous agent to generate its behavior. In particular, in Section 5.2.1 we specify the structure of such representation and we connect it to the notion of spatio-temporal affordance. In Section 5.2.2, instead, we describe how the Spatio-Temporal Affordance Map is a task-directed representation.

5.2.1. Structure and Specifications

A STA can be learned, updated and used by an autonomous agent, through a Spatio-Temporal Affordance Map (STAM). STAM is a modular representation (see Figure 5.2 and Figure 5.3), whose core elements are the spatio-temporal affordance

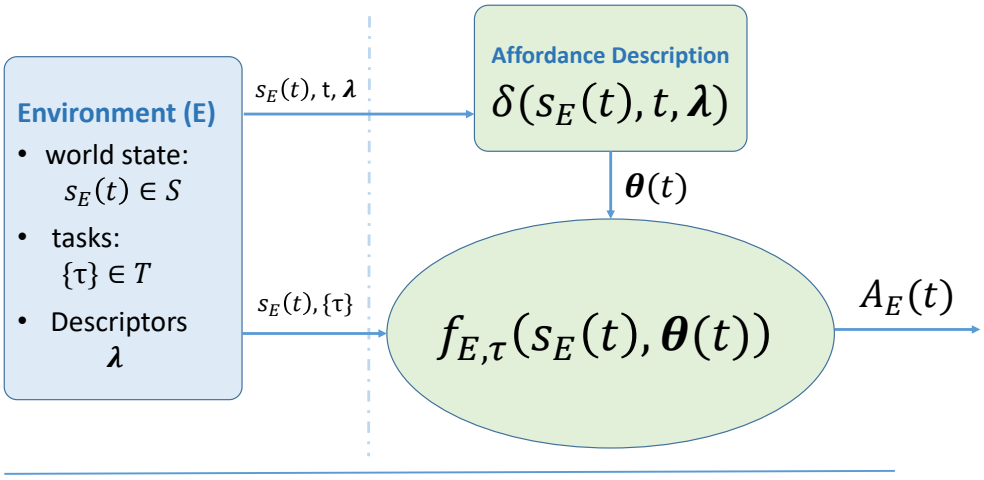


Figure 5.2. Schema of a Spatio-Temporal Affordance Map.

$f_{E,\mathcal{T}}$ and the affordance description function δ_E. These functions are interconnected with the world by means of the "environment module", which stores the affordance descriptors $\boldsymbol{\lambda}$, the current state of the world s_t and the set of tasks $\mathcal{T}$ that are defined in the environment.

Definition 5.3. *A **spatio-temporal affordance map** is a pair*

$$\langle a_m, e_m \rangle,$$

where:

- a_m *is an affordance module, in which the spatio-temporal affordance* $f_{E,\mathcal{T}}$ *and the affordance description function* δ_E *are applied;*
- e_m *is an environment module, that stores, at each time* t *the state* s_t *of the environment, the descriptors* $\boldsymbol{\lambda}$ *and the tasks* $\{\tau_j\}$ *defined in* E*.*

The spatio-temporal affordance map is the interconnection between the real environment and the STA, which instantiates $f_{E,\mathcal{T}}$ and δ_E to a specific representation of the environment E. Throughout this thesis, we assume a semantic map, as described in Chapter 4, to be part of the environment module of a STAM and to provide the (semantically relevant) state of the world to the STA function. The relation between semantic maps and STAM also supports the revision of knowledge for decision making, expressed in the form of spatio-temporal affordances. In particular, this is possible thanks to the structure of the basic concept hierarchy (Definition 4.1) that introduces a predicate *hasAffordance* and a class `Affordance`, directly mapped in the environment. The class `Affordance` is instantiated in the form of STA signatures, that can be modified or used when some conditions occur, or as a function of the knowledge that is stored in the semantic map.

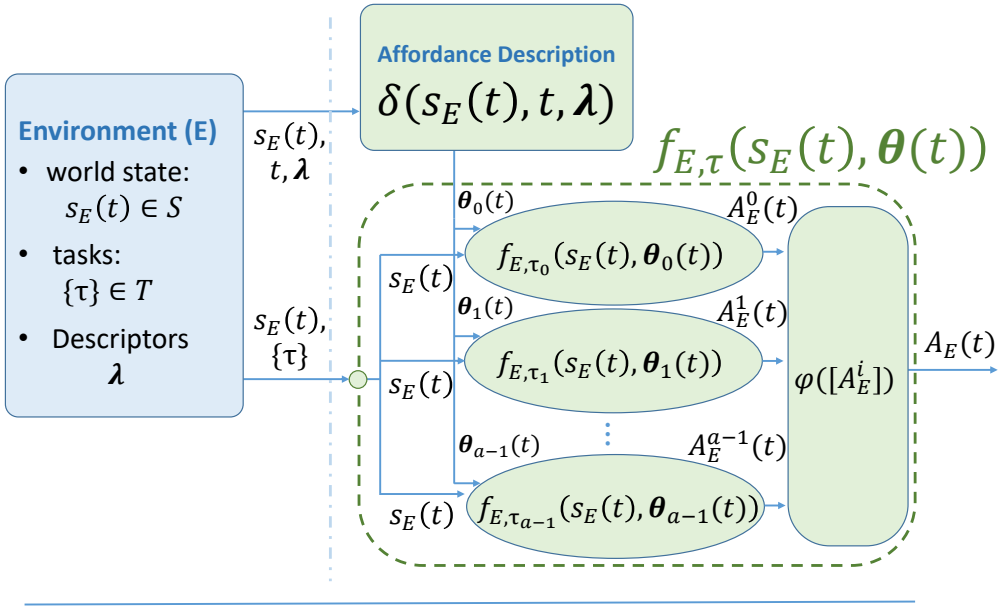


Figure 5.3. Decomposition and detailed overview of a Spatio-Temporal Affordance Map.

Remark 5.7. *A semantic map, as introduced in Definition 4.3, can store the parameters $\boldsymbol{\lambda}$ of a spatio temporal affordance and can be used as a part of the environment module of STAM.*

5.2.2. Action Semantics and Task-Directed Representations

Spatio-temporal affordances are intended to directly influence the behavior of an autonomous agent, and to affect both situated action and decision making. At each time, while determining its own behavior, an agent needs to consider the effects of its possible actions and to choose the most promising one.

Definition 5.4. *In the context of decision making, the term **action semantics** refers to the implicit or explicit evaluation of effects of an action.*

Definition 5.5. *A **task-directed representation** is defined as a representation that characterizes an environment E in terms of its action semantics.*

Intuitively, a spatio-temporal affordance encodes the semantics of an action α by expressing its "goodness", given the current state of the environment. To do so, it implicitly evaluates the effects of α to predict how beneficial its execution is to the purposes of the agent. For this reason, a spatio-temporal affordance map is generated as a task-directed representation, that models the environment through its action semantics, which is implicitly encoded by means of affordances.

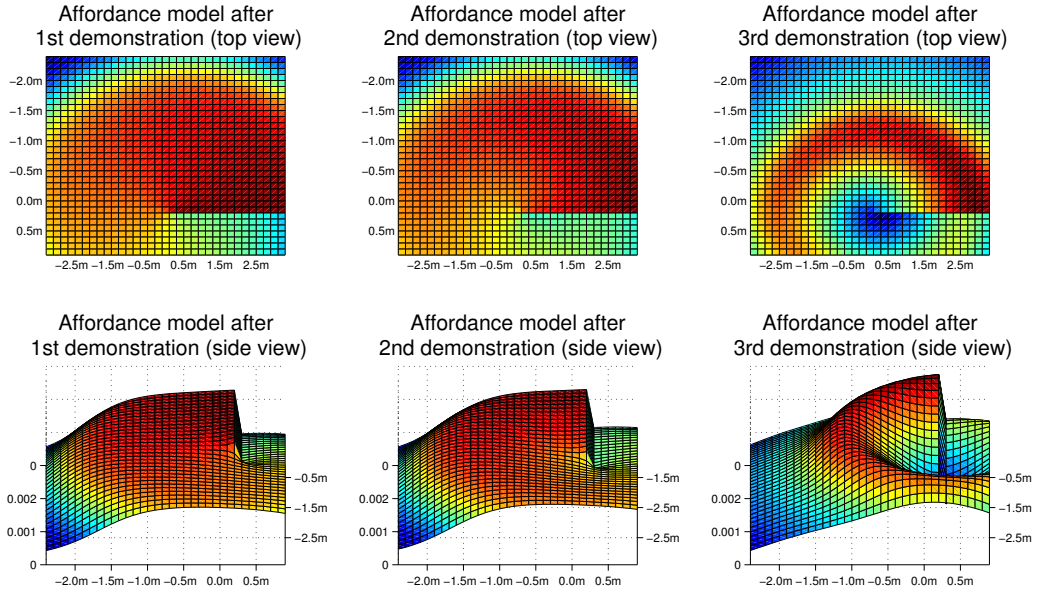


Figure 5.4. Spatio-temporal affordance of a following task, at a given time t, learned with increasing number of expert demonstrations. Here, the target is located at the origin and the plots represent the probability density function of a pose to afford the task. The plots, whose coordinates are expressed in meters, show that the model is able to represent both minimum and maximum distances from the target, in accordance with the data provided as demonstrations.

5.3. Explanatory Use Case

In order to illustrate our model, in this section we introduce an explanatory use case, where a robot R is assigned the task of following a person P and we want to learn the corresponding spatio-temporal affordance function.

In this situation, the areas of the environment E that afford the task depend on manifold factors, such as general rules (e.g., forbidden areas), user preferences (that can be encoded in the set of parameters $\boldsymbol{\theta}_t$) and the position of the followed person (encoded in the state of the environment s_t). According to Definition 5.1, however, we can use a spatio-temporal affordance function for generating a map A_E, that identifies the poses of R which afford the assigned task, given a state of the environment s_t. Such state is encoded, at each time t, by the positions x, y and the orientation α

$$\langle x_{t,P}, y_{t,P}, \alpha_{t,P}, x_{t,R}, y_{t,R}, \alpha_{t,R} \rangle$$

of the target P and the robot R.

At learning time, we use a Gaussian Mixture Model (GMM) and Expectation Maximization (initialized with the k-means algorithm) for optimizing Eq. (5.3). The signatures $\boldsymbol{\theta}_t$ of the STA function are then defined as a tuple

$$\boldsymbol{\theta}_t = \langle \pi_{t,1}, \mu_{t,1}, \Sigma_{t,1}, \ldots, \pi_{t,N}, \mu_{t,N}, \Sigma_{t,N} \rangle,$$

where $\pi_{t,i}$ is the prior, $\boldsymbol{\mu}_{t,i}$ the mean vector and $\boldsymbol{\Sigma}_{t,i}$ the covariance matrix of a mixture of N Gaussians, at each time t. In this simple experiment, we set $N = 2$ and the affordance signatures $\boldsymbol{\theta}_t$ are learned from datasets $\mathcal{D}_t$ collected through multiple demonstrations of different experts. The affordance descriptors, instead, are learned by means of a simple Ridge Regression.

To collect expert data, we setup two robots in a simulated environment – one randomly roaming through the environment and simulating P, the other being controlled by an expert through a joystick and following the target P. During these sessions, the state s_t, as defined above, is recorded at each time instant. The learned model is finally used by the robot R to determine a probability map (illustrated in Figure 5.4) of the best follower pose, given the pose of P. In this example, no specific constraint is imposed on the robot for the selection of its path. Hence, the agent can select the pose that greedily maximizes its utility over the probability map and reach it by following the shortest path.

5.4. Contributions

In this chapter, we discussed our model of Spatio-Temporal Affordance Maps (Def 5.3), that is based on the notion of spatio-temporal affordances (Definition 5.1). This concept, builds on ecological theories and extends previous approaches by directly connecting the environment, its spatial semantics, and their evolution. Practically, this connection can be realized by means of our model of semantic map, and is useful to the purposes of behavior generation. In fact, in Chapter 7, we directly use spatio-temporal affordances to generate effective robot policies.

The main contributions introduced in this chapter consist in (1) the formalization of a novel model that is both time-invariant and time-dependent, and that directly represents action semantics in relation to the space of the environment, by encoding the desirability or value of the actions; (2) the definition of a procedure for learning such representation, given example data; (3) the description of a representation that, for the first time, directly connects the notion of spatio-temporal affordances with the concept of semantic maps. In particular, this chapter reports and extends our prior work [116] by enhancing STAM to the time-dependent setting, by establishing the properties of the model and the procedure for its optimization, and associating the model to the semantic map representation.

Part III

Algorithms

Chapter 6

Online Semantic Mapping

The best educated people are those who are always learning, always absorbing knowledge from every possible source and at every opportunity.

— Orison Swett Marden

We begin the exploration of algorithms for the generation of semantic-driven robot behaviors by focusing our attention on deliberative systems, which are directly interconnected with a semantic map adhering to the formalization in Chapter 4. In particular, in this chapter, we introduce our approach for gathering semantic information and generating high-level adaptive behaviors at run-time, given the current state of the environment. While collecting information about the world, we enable our robots to interact both with the environment and with humans, and to proactively infer novel information from such interaction. Then, the robot uses the newly collected information to improve its own behavior and degree of autonomy. In accordance with the statement of this thesis, and the model developed in Chapter 4, we directly leverage traditional reasoning and inference mechanisms for generating high-level behaviors by means of semantic maps.

The chapter is organized as follows. First, we describe our algorithm for gathering semantic information online (Section 6.1), as well as for exploiting interactions with humans and with the world (Section 6.1.1). Then, in Section 6.2, we illustrate how this is used both for inference (Section 6.2.1) and for online generation of robot behaviors (Section 6.2.2). Finally, in Section 6.3 we discuss the experimental evaluation of the approach.

6.1. Online Information Acquisition

We now consider an incremental procedure for knowledge acquisition under the formalization of Chapter 4. As established in Definition 4.6, an incremental semantic mapping process is a function that, at each time-step t, takes as input the semantic map at $t-1$ and some data from the world (e.g., sensor data, events), and emits a $\mathcal{SM}$ representation that includes the newly recorded information. Under this perspective, in this section we develop an algorithm – **online semantic mapping** –

Input: $\mathcal{SM}_I$: initial semantic map.
Output: $\mathcal{SM}_F$: final semantic map.
Data: $\mathcal{M}$: set of perceptions; $\mathcal{E}$: set of recorded events; $\mathcal{C}$: set of commands.
begin
  Initialize $\mathcal{SM}_F \leftarrow \mathcal{SM}_I$.
  while *executing()* **do**
    $\mathcal{SM}_{t-1} \leftarrow \mathcal{SM}_F$
    `// Use sensors and perceive the world`
    $\mathcal{M}_t \leftarrow$ sense()
    `// Parse sensor data by using` $\mathcal{SM}$ `to generate events`
    $\mathcal{E}_t \leftarrow$ parse($\mathcal{M}_t$, $\mathcal{SM}_{t-1}$)
    $\mathcal{C}_t \leftarrow$ getCommands($\mathcal{E}_t$)
    `// Store events after extracting commands`
    $\mathcal{E}_t \leftarrow \mathcal{E}_t \setminus \mathcal{C}_t$
    if $|\mathcal{E}_t| > 0$ **then**
      `/* Generate a new semantic map according to: the previous map, the perceptions and the events */`
      $\mathcal{SM}_t \leftarrow \phi_{ISM}(\mathcal{M}_t, \mathcal{E}_t, \mathcal{SM}_{t-1})$
    end
    `// Act according to new semantic map and commands`
    act($\mathcal{SM}_t$,$\mathcal{C}_t$)
    $\mathcal{SM}_F \leftarrow \mathcal{SM}_t$
  end
  return $\mathcal{SM}_F$
end

Algorithm 1: Pseudo-code for Online Semantic Mapping.

that performs incremental semantic mapping at run-time, through the interaction of the robot with the environment and the humans in it. In its general form, the pseudo-code of this algorithm is very simple, and it is illustrated in Algorithm 1. During execution, in fact, the robot first uses its sensors to perceive the surrounding world and, then, it parses incoming data together with previous knowledge in $\mathcal{SM}$ to generate a set of events $\mathcal{E}$. Hence, after discerning between commands and simple events, the agent uses $\mathcal{E}$ to update the semantic map $\mathcal{SM}$. Finally, it acts according to the updated semantic map and to the received commands, if any.

In the remainder of this section, we first make explicit the function that we use for parsing perceived data (Section 6.1.1) and, then, we detail our implementation of ϕ_{ISM} (Section 6.1.2). The process for semantic-driven behavior generation, instead, is illustrated in the next section (Section 6.2).

6.1.1. Sensor Data Parsing

As illustrated in Algorithm 1, our semantic mapping procedure relies on a set of events $\mathcal{E}$ that are generated after parsing sensor data $\mathcal{M}$. Importantly, at each time t, these events not only depend on sensory information, but also on previous knowledge stored in the semantic map at $t-1$. This means, for example, that until an object has not been learned and memorized in $\mathcal{SM}$, the robot is neither able to categorize it, nor to generate an event corresponding to its recognition. Intuitively, this enables the robot to primarily memorize information that is useful for its purposes, or for

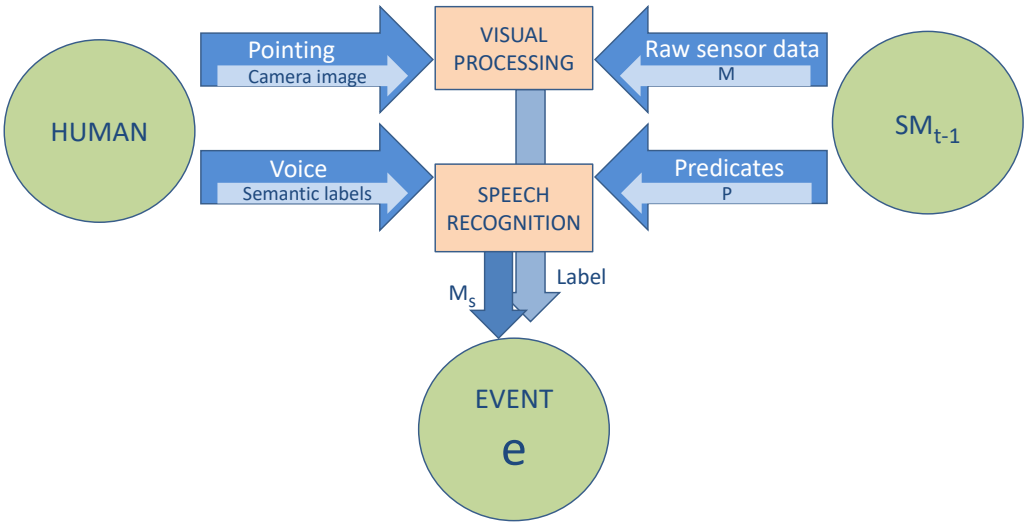


Figure 6.1. Event (label and subset of sensor data) generated through human-robot interaction.

Input: V: human voice; I: camera image; $\mathcal{SM}$: available semantic map.
Output: e: event.
Data: $\mathcal{P}$: set of predicates stored in $\mathcal{SM}$; $\mathcal{M}$: set of raw sensor data stored in $\mathcal{SM}$.
begin
  $(\_, \mathcal{M}, \mathcal{P}) \leftarrow \mathcal{SM}$
  $L \leftarrow \text{speechRecognition}(V, \mathcal{P})$
  $\mathcal{M}_s \leftarrow \text{visualProcessing}(I, \mathcal{M})$
  $e \leftarrow (L, \mathcal{M}_s)$
  return e
end

Algorithm 2: Pseudo-code for event generation in the case of human-robot interaction.

Input: I: camera image; $\mathcal{SM}$: available semantic map.
Output: e: event.
Data: $\mathcal{P}$: set of predicates stored in $\mathcal{SM}$; $\mathcal{M}$: set of raw sensor data stored in $\mathcal{SM}$.
begin
  $(\_, \mathcal{M}, \mathcal{P}) \leftarrow \mathcal{SM}$
  $\mathcal{M}_s \leftarrow \text{visualProcessing}(I, \mathcal{M})$
  $L \leftarrow \text{objectRecognition}(\mathcal{M}_s, \mathcal{P})$
  $e \leftarrow (L, \mathcal{M}_s)$
  return e
end

Algorithm 3: Pseudo-code for event generation in the case of automated object recognition.

the purposes of its user.

In our algorithm, the set $\mathcal{E}$ can be obtained from different sensor sources, that during execution are employed by the robot both for human-robot interaction, and

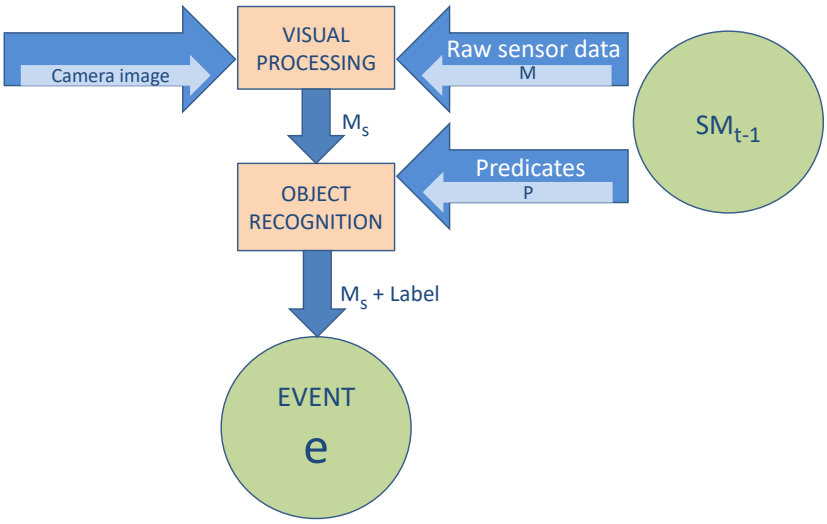


Figure 6.2. Event (label and subset of sensor data) generated through autonomous object recognition.

for autonomous object recognition. Concretely, after parsing sensory information, these events consist of either labels associated to a set $\mathcal{M}_s \subseteq \mathcal{M}$ of semantically relevant sensor data, or commands that are represented as strings.

Remark 6.1. *An event consists of labels associated to a set $\mathcal{M}_s \subseteq \mathcal{M}$ of semantically relevant sensor data.*

Following Remark 6.1, and without considering received commands, in the case of human-robot interaction (Figure 6.1) a single event e is generated as illustrated in the pseudo-code of Algorithm 2. In particular, we assume a human supports the robot in knowledge acquisition by pointing to an object while vocally labeling it [43]. In the case of automated object recognition (Figure 6.2), instead, an event is generated as in the pseudo-code of Algorithm 3. Importantly, each function for visual processing, speech recognition and object recognition – whose implementation may vary depending on the application and domain – considers semantic knowledge coming from a semantic map $\mathcal{SM}$ and represented according to Def 4.3. Thanks to this, the robot can either understand when new knowledge is provided, in the case of human-robot interaction, or guide the search of already known objects in a perceived scene. For instance, let us assume an environment E with three platforms, all of them represented in a semantic map $\mathcal{SM}$, without further details. If a human labels one of the platforms as the "yellow platform", then the speech recognition can generate a new event according to the fact that "yellow" was not stored in $\mathcal{SM}$.

Once events are created, as described above, they need to be processed by means of ϕ_{ISM} to be stored in a semantic map representation. This activity is described in the next section.

Input: $\mathcal{M}_t$: set of perceptions; $\mathcal{E}_t$: set of events; $\mathcal{SM}_{t-1}$: initial semantic map.
Output: $\mathcal{SM}_t$: updated semantic map.
begin
  `// Get the elements of the semantic map at` $t-1$
  $(R, \mathcal{M}_{t-1}, \mathcal{P}_{t-1}) \leftarrow \mathcal{SM}_{t-1}$
  `// Align new sensor data to the reference frame of the map`
  $\mathcal{M}_t \leftarrow \text{align}(R, \mathcal{M}_t, \mathcal{M}_{t-1})$
  `// Merge the aligned sensor data`
  $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup \mathcal{M}_{t-1}$
  `// Get elements of the event`
  $\{(L, \mathcal{M}_s)_i\}_t \leftarrow \mathcal{E}_t$
  `// Generate a predicate depending on labels and prior knowledge`
  $\mathcal{P}_t \leftarrow \text{generatePredicate}(L_t, \mathcal{P}_{t-1})$
  `// Check if new predicates are consistent with old ones`
  $\mathcal{P}_{TMP} \leftarrow \mathcal{P}_t \cup \mathcal{P}_{t-1}$
  if not *consistent*$(\mathcal{P}_{TMP})$ **then**
    `/* adopt recovery behavior, if desired, by asking the help of the user to manage consistency */`
    return
  end
  `// If predicates are consistent, add them`
  $\mathcal{P}_t \leftarrow \mathcal{P}_{TMP}$
  `// Approximate spatial information in predicates`
  $\mathcal{P}_{s,t} \leftarrow \text{approximate}(\mathcal{M}_{s,t})$
  `// Add spatial predicates`
  $\mathcal{P}_t \leftarrow \mathcal{P}_t \cup \mathcal{P}_{s,t}$
  `// Update the semantic map`
  $\mathcal{SM}_t \leftarrow (R, \mathcal{M}_t, \mathcal{P}_t)$
  return $\mathcal{SM}_t$
end

Algorithm 4: Pseudo-code for the incremental semantic mapping function ϕ_{ISM}.

6.1.2. Semantic Map Generation

In this section, we discuss the implementation of ϕ_{ISM} (Algorithm 4) and we connect it to the structural bias problem by means of an example. In particular, given a semantic map $\mathcal{SM}_{t-1}$, perceptions $\mathcal{M}_t$ and events $\mathcal{E}_t$, at time t, we aim at generating a $\mathcal{SM}_t = (R, \mathcal{M}_t, \mathcal{P}_t)$ where:

- $\mathcal{M}_t$ is expressed in the reference frame $R \in \mathcal{SM}_{t-1}$ and is integrated with the raw sensor data $\mathcal{M}_{t-1} \in \mathcal{SM}_{t-1}$ as $\mathcal{M}_t = \mathcal{M}_t \cup \mathcal{M}_{t-1}$;
- predicates $\mathcal{P}_t$ are obtained from the events $\{e_i\}_t = \{(L, \mathcal{M}_s)_i\}_t$, and their consistency is verified against the predicates $\mathcal{P}_{t-1} \in \mathcal{SM}_{t-1}$;
- predicates $\mathcal{P}_{s,t} \in \mathcal{P}_t$ are generated to represent semantically relevant sensor data $\mathcal{M}_{s,t} \in \mathcal{M}_t$.

The expression of $\mathcal{M}_t$ in the frame R, and its merging $\mathcal{M}_t = \mathcal{M}_t \cup \mathcal{M}_{t-1}$, can considered as general registration (or alignment) problem, on which a vast literature

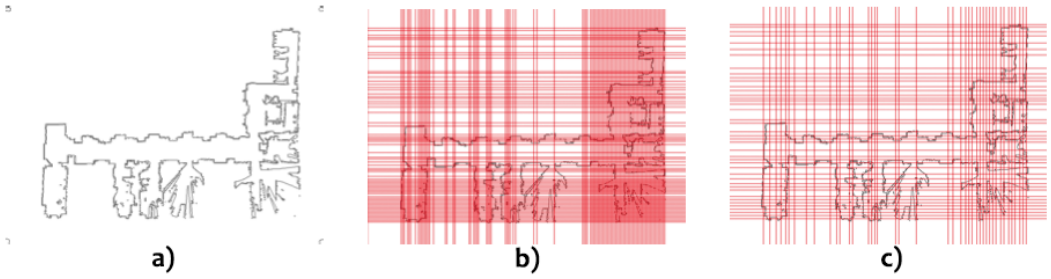


Figure 6.3. Steps of the construction of the A-Grid [20], given a metric map.

exists – and that can by solved, for example, through reliable approaches such as the Iterative Closest Point algorithm [11] and its derived methods [129, 61, 131].

The generation of meaningful predicates can be obtained via a parser or, more simply, through string matching between the labels L_t of $\mathcal{M}_{s,t}$, and the classes expressed in the concept hierarchy of the semantic map. For example, when $\mathcal{M}_{s,t}$ is labeled as "object" (or any specialization of it), this can be easily matched with the class `Object` belonging to $\mathcal{SM}$. When no match is found, either additional interactions with a human can be triggered, or the information can be discarded. The consistency of the newly obtained predicates can be simply verified by using a temporary set $\mathcal{P}_{TMP} = \mathcal{P}_t \cup \mathcal{P}_{t-1}$, together with a reasoner for the language representing the concept hierarchy. In the case in which $\mathcal{P}_{TMP}$ is consistent, we can set $\mathcal{P}_t = \mathcal{P}_{TMP}$. Otherwise, as before, an interaction can be triggered, or the information can be discarded.

Finally, to express $\mathcal{M}_{s,t}$ by means of predicates $\mathcal{P}_{s,t}$, an approximation function can be used. This function is supposed to introduce a discretization of raw sensor data in $\mathcal{SM}$, with the goal of representing them by means of logical predicates – but without loss of spatial reference. Although there are several viable alternatives, such as bounding boxes, KD-trees or quadtrees, in the next paragraph we provide an example of such function for 2D points represented in a metric map.[1]

Example of Approximation Function for 2D Metric Maps We introduce an approximation of spatial information named A-Grid [20], that can be used for discretizing 2D metric maps. While details regarding its construction are out of the scope of this chapter, and can be found in Capobianco et al. [20], here we shortly discuss some of the consequences that derive from its use. In particular, the A-Grid is a non-uniform grid whose cells vary as a function of the occupied areas of the map (Figure 6.3), and whose compression rate with respect to the original metric map is $\sim 98.5\%$ [43]. As such, this approximation can be easily and efficiently accessed as a matrix, by means of its indices. Consequently, it can be represented through predicates $\mathcal{P}_s$ that store its indexing. The A-Grid introduces an average structural bias that is estimated as $57\% \pm 18.5$ [43] with respect to the real size of the represented spatial element. This value can be simply computed by: (1) manually inserting ground truth information in $\mathcal{SM}$, to discard sensor errors and

[1]Note that a metric map satisfies the requirement for $\mathcal{M}$ in Definition 4.2 of preserving raw sensor data, since range information can be reconstructed from it, and it is represented in a specific reference frame R.

obtain predicates $\mathcal{P}_s$; (2) back-projecting spatial information from the predicates $\mathcal{P}_s \in \mathcal{SM}$ to the metric space; (3) comparing the obtained spatial properties (e.g., the size) against the physical ground truth. Hence, an object memorized in $\mathcal{SM}$ through the A-Grid is represented, on average, almost 1.5 times bigger than its real size.

The measure of the structural bias of a semantic mapping approximation function, is a key step towards the design of a robotic system. In fact, depending on the domain of application, a certain approximation may or may not be suitable for a robot. In the prototype applications of this thesis (Section 6.3.2), for example, we adopt the A-Grid because of its trade-off between efficiency and structural bias. In fact, despite approximation errors at deliberation levels, for situated action we leverage affordance-guided behaviors, that are not influenced by the predicates $\mathcal{P}_s$.

6.2. Inference and Robot Behaviors

In previous sections, we introduced an approach that enables a robot to store knowledge in a semantic map at run-time. Here, we discuss how this information can be used for inference, reasoning, and to generate high-level robot behaviors. In particular, we explain how tasks can be executed by a robot in accordance to both specified commands and semantic knowledge in $\mathcal{SM}$. Under these settings, in fact, inference and spatial reasoning can be used, at each time t, to understand and ground users' commands with respect to the modeled semantic map $\mathcal{SM}_t$.

In this section, we first describe how $\mathcal{SM}_t$ can be used to interpret commands through inference and spatial reasoning (Section 6.2.1) and, then, we use these concepts to evaluate their effects on autonomous agent's behavior (Section 6.2.2).

6.2.1. Command Disambiguation, Inference and Spatial Reasoning

According to Merriam-Webster, disambiguation is the process of establishing a single semantic or grammatical interpretation for a given word or sentence. In our context, users' commands are directly grounded to known concepts and scenarios – i.e., the semantic map. In fact, disambiguation in a robotic system is essential to address the typical uncertainty of natural language, as well as to achieve effective behaviors given ambiguous commands. Usually, ambiguity arises when multiple concepts or objects are available in the environment, with equal names and functionalities (e.g., a platform), but different properties (e.g., the color of the platform) or specializations (e.g., platform holding slotted screwdrivers). Under these circumstances, humans flawlessly distinguish objects and situations through properties, specializations and spatial relations. In this section, we explain how these mechanism can be implemented in a robot by means of a semantic map, as well as how they can be used to accumulate additional knowledge at run-time.

Inference Let us consider an environment E that, exactly as Sandy's world (Section 1.1), is composed by three platforms, each of them storing different objects. Moreover, let us assume an initial setting in which a semantic map $\mathcal{SM}_E$ is available to the robot. In this map, the notions of `Platform`, `Box` and `Slotted_Screwdriver`

Input: $\mathcal{SM}_t$: current semantic map; $\mathcal{C}_t$: received commands.
Output: π: planned actions.
Data: $\mathcal{P}_G$: set of grounded predicates; $\mathcal{P}_{NG}$: set of non-grounded predicates.
begin
  // Get the elements of the semantic map at time t
  $(R, \mathcal{M}_t, \mathcal{P}_t) \leftarrow \mathcal{SM}_t$
  // Extract predicates corresponding to commands
  $\mathcal{P}_c \leftarrow$ parseCommands$(\mathcal{C}_t)$
  // Evaluate predicates that can or cannot be grounded (e.g., through a reasoner and QSR)
  $(\mathcal{P}_G, \mathcal{P}_{NG}) \leftarrow$ ground$(\mathcal{P}_c, \mathcal{P}_t)$
  // Check whether some commands have not been grounded
  if $|\mathcal{P}_{NG}| > 0$ **then**
    // Check whether (1) non-grounded predicates add knowledge and (2) are consistent with available knowledge
    $\mathcal{P}_{TMP} \leftarrow \mathcal{P}_{NG} \cup \mathcal{P}_t$
    if not *consistent*$(\mathcal{P}_{TMP})$ **then**
      /* adopt recovery behavior, if desired, by asking the help of the user to manage consistency */
    else
      // If non-grounded predicates add knowledge and are consistent, integrate them in the semantic map
      $\mathcal{P}_t \leftarrow \mathcal{P}_{TMP}$
      $\mathcal{SM}_t \leftarrow (R, \mathcal{M}_t, \mathcal{P}_t)$
    end
  end
  // Request a plan from the reasoner, according to the commands and the available semantic map
  $\pi \leftarrow$ *plan*$(\mathcal{C}_t, \mathcal{SM}_t)$
  // if $\mathcal{SM}_t$ does not contain enough information, the reasoner cannot infer a plan
  if $\pi = \emptyset$ **then return**;
  return π
end

Algorithm 5: Pseudo-code for online inference, given the specification of a command.

are represented, but no further instantiation is provided, besides the presence in E of (1) three platforms (A, B and C) and (2) a box on platform C. At time t_{c_1}, the robot receives a command to "take a slotted screwdriver from the box on platform A". This command is well detailed, with no ambiguity and multiple specifications that clarify the request and specialize the robot's knowledge. In this case, the robot can "easily" ground and understand the request of the human. Moreover, it can also store in $\mathcal{SM}_E$ additional information in the command – i.e., the presence of a box on platform A, that stores slotted screwdrivers. This is accomplished by means of new predicates $\mathcal{P}_{NEW}$. Hence, at any time $t > t_{c_1}$, the semantic map $\mathcal{SM}_E$ contains a set of predicates $\mathcal{P} = \mathcal{P}_{NEW} \cup \mathcal{P}_{t_{c_1}}$ that represent the robot's knowledge of the environment. When at time $t_{c_2} > t_{c_1}$ the robot is requested to "put the slotted

screwdriver in the box", the command, in order to be executed, requires the inference of additional knowledge from the semantic map. In fact, since multiple boxes are located in the environment, the request is ambiguous and cannot be immediately satisfied. However, by using the information learned at t_{c_1} and a reasoning engine, the robot is able to execute the task.

We illustrate the process discussed in our example in Algorithm 5. For this algorithm, we assume the availability of a command parser that is able to generate a set of meaningful predicates $\mathcal{P}_c$, given a command $c \in \mathcal{C}$.

Qualitative Spatial Reasoning To correctly interpret commands, and tackle the ambiguity of multiple instances of a class, spatial relations can be used, given a semantic map. Let us consider, for example, the same scenario as before. In this case, however, in addition to platforms and boxes, the semantic map also contains the concept of `Wall`, together with its corresponding instances for the environment E. Additionally, although not explicated by predicates in $\mathcal{SM}_E$, A is the only platform situated in the vicinity of a wall. Under these settings, at a certain time t, the robot is given the command "go to the yellow platform next to the wall". Since the exact label of the platform is not given, the command is ambiguous and its execution is not straightforward. In particular, to satisfy its request, the agent must evaluate spatial relations between elements in $\mathcal{SM}_E$ – and infer the goal of the task (A) through its vicinity to a wall. To do so, the robot can use the predicates $\mathcal{P}_s$, and apply a reasoner as the one based on the model in Figure 6.4. As in the previous example, the robot can store new knowledge in the map – i.e., the color of the platform – thanks to the grounded information – i.e., the goal platform A. Although this process is not explicated in Algorithm 5, qualitative spatial reasoning can be used as a part of the grounding function.

6.2.2. Online Use of Semantic Knowledge in Robot Behavior

Once a semantic map $\mathcal{SM} = (R, \mathcal{M}, \mathcal{P})$ is available, and situated behaviors are accessible through descriptors $\boldsymbol{\lambda}$ in the `Affordance` class of $\mathcal{P}$, these can be used by a robot for planning and deliberative behavior generation. In this setting, we define "complex tasks", a set of situated behaviors that is combined through specific operators. A complex plan can be generated by means of a generic planner, whose domain knowledge includes instances and classes belonging to $\mathcal{P} \in \mathcal{SM}$.

After a plan is given, deliberated behaviors can be finally executed. To this purpose, although alternative formalisms might be equivalently valid, we leverage Petri Net Plans. As introduced in Section 2.3, a PNP can express non-instantaneous actions and conditions that are verified at run-time. This effectively meets our needs of deliberated action control, given situated behaviors that are implemented through affordance based decision making. A Petri Net Plan, generated for the screwdriver pick-up task of previous examples, is given in Figure 6.5. Here, the robot executes two parallel activities. The former consists in visiting all the instances of the class `Platform` – A, B and C. This enables the robot to look for the desired screwdriver and execute the pick-up action through appropriate affordance descriptors. The latter, instead, consists in using sensory data to determine whether the screwdriver can be picked-up.

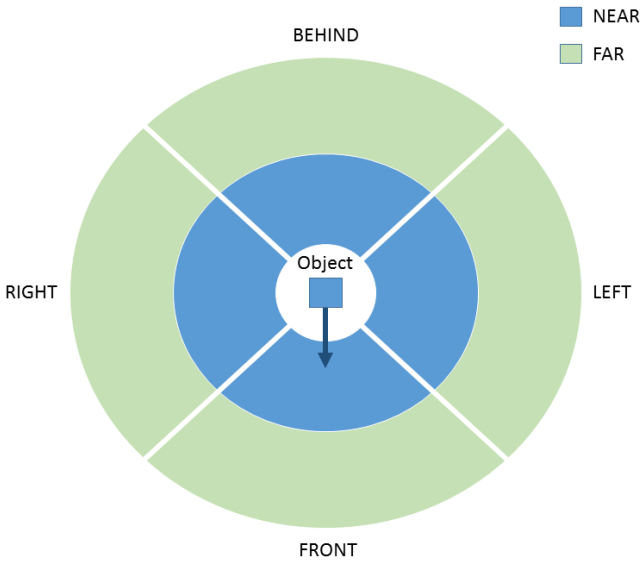


Figure 6.4. Example of qualitative spatial relations. Here, relative directions are computed against the intrinsic normal vector of an object, while distances are highlighted with circles of different colors.

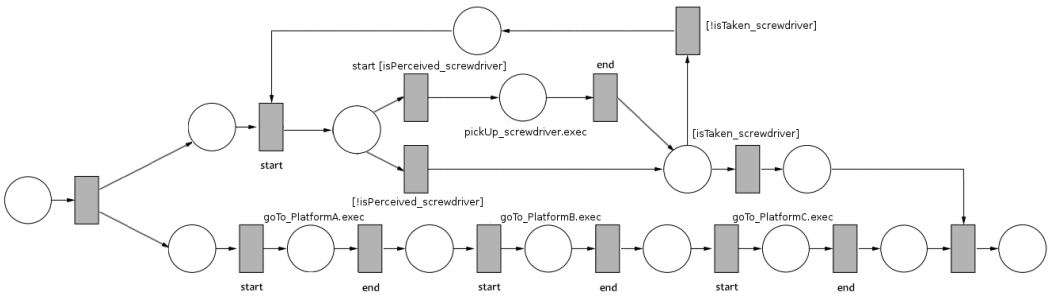


Figure 6.5. Example of PNP for visiting platforms and picking-up a screwdriver.

6.3. Evaluation Challenges and Experiments

The experimental evaluation of the online semantic mapping algorithm, as proposed in this chapter, necessarily needs to be performed on specific instances of the perception, parsing, approximation and planning functions. These, however, introduce errors that can be measured, but whose effects exponentially concatenate within the full semantic mapping system. Due to them, this type of evaluation does not allow to effectively measure the method itself, in terms of generated behaviors. While we evaluate an instance of the online semantic mapping algorithm in Section 6.3.2, we first propose a thought experiment[2] that attempts to address this issue (Section 6.3.1).

[2]A thought experiment is a test in which one imagines the practical outcome of a hypothesis, action or condition.

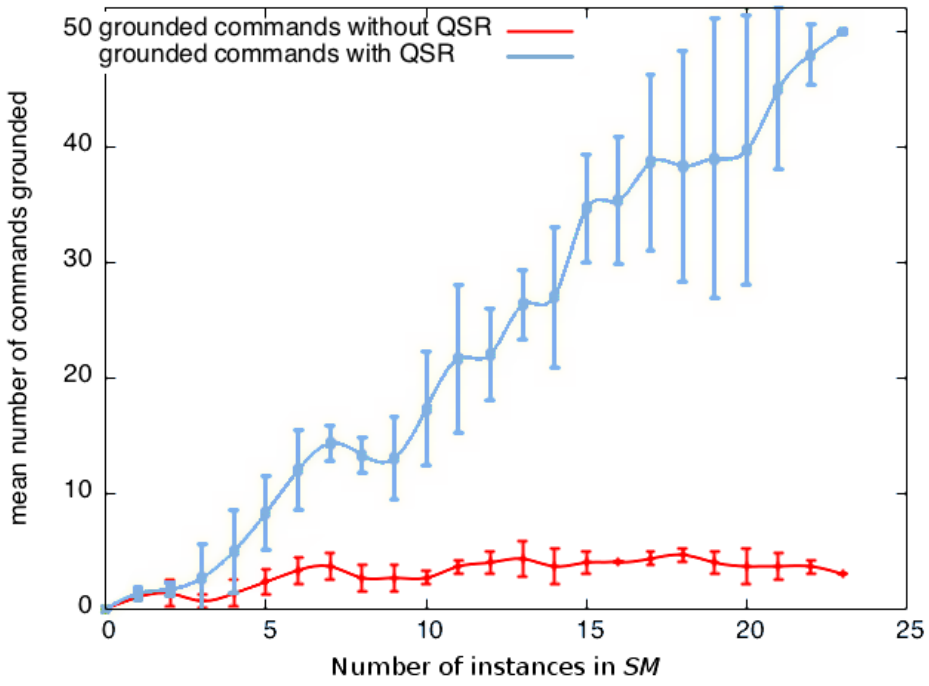


Figure 6.6. Number of grounded commands, given a certain number of concept's instances in SM.

6.3.1. Thought Experiment

Our thought experiment articulates as follows and relies on six assumptions:

1. the use of an initial empty semantic map $\mathcal{SM}_I$, that is coherent with our model (Definition 4.3) and implements a basic concept hierarchy;

2. the accessibility to an oracle that, at each time instant $t = 0 \dots 23$[3], chooses among 25 different instances belonging to one of the classes `Location`, `Object`, `Connecting_Architecture` and generates an artificial event e_i;

3. the availability of 50 different (95% ambiguous and 5% non ambiguous) commands, that at each time are provided to the robot;

4. the accessibility to a perfect generator for predicates, given events and commands e_i;

5. the accessibility to a qualitative spatial reasoner, that acts as illustrated above and follows the model in Figure 6.4;

6. the correct grounding of a command implies that there is no error in the generation of a valid plan for the robot.

[3]Note that, since $t = 0 \dots 23$, two random instances are left out of this procedure to generate a less biased estimate.

In particular, assumptions 2, 3 and 4 guarantee an unbiased evaluation of our method, that does not rely on error prone functions. Similarly, assumption 6 ensures our experiment to be independent from eventual planning faults. Given these premises, the thought experiment proceeds by using the oracle to repeatedly fill the semantic map $\mathcal{SM}_t$, at each time instant t, through a function ϕ_{ISM} that makes use of the perfect predicate generator. Then, $\mathcal{SM}_t$ is used through Algorithm 5 (1) to process each of the 50 commands both with and without the use of the available qualitative spatial reasoner, and (2) to measure the number of correctly grounded commands – i.e., successful task executions, according to assumption 6.

We manually performed this activity repeatedly, starting from scratch five times and generating the graph illustrated in Figure 6.6. From the graph, both the impact and the variance of the integrated qualitative spatial reasoner emerge. The latter depends on the random order of knowledge insertion, as well on the 2 random instances that are left out of the map (assumption 2) at each re-iteration of the procedure. As expected, instead, the mean and variance of the non-QSR approach account for about the $5\% \pm 1$ of successful grounded commands – i.e., the number of non ambiguous commands.

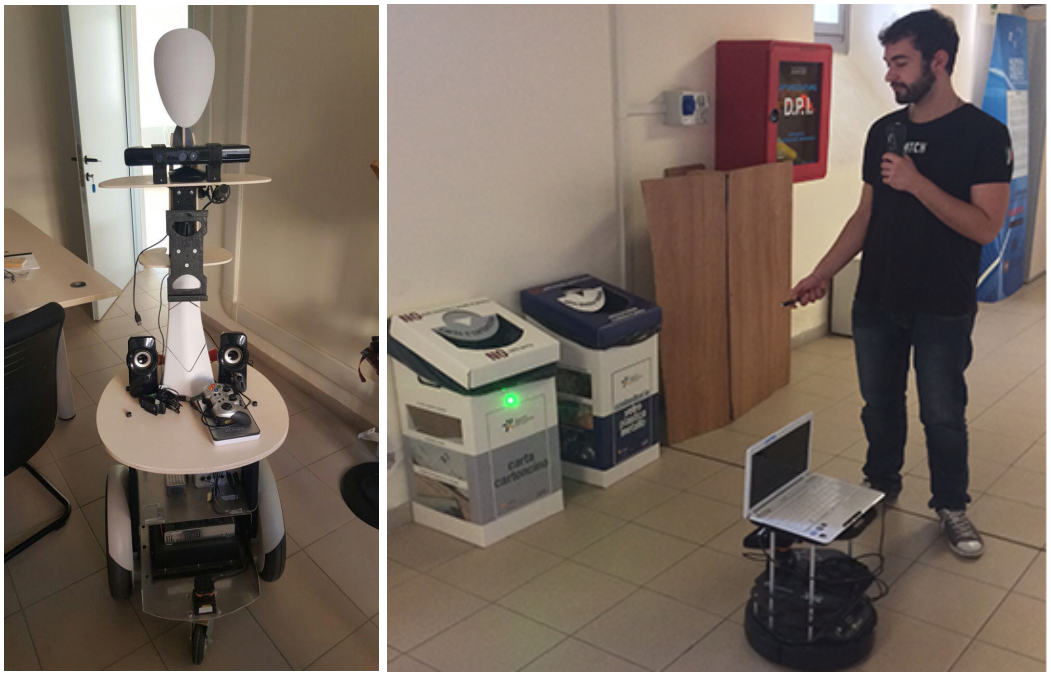
The thought experiment confirms our insights about the necessity of semantic reasoning capabilities for successful deliberation in robot behaviors. Additionally, the experiment illustrates how a robot, endowed with an online semantic mapping procedure, can increase its autonomy and adaptation capabilities almost linearly with respect to the amount of knowledge it has learned. Importantly, these insights are substantially confirmed by our experiments on prototype applications (discussed in Section 6.3.2), meaning that effective behavior can be practically obtained through our approach.

6.3.2. Semantic-Driven Service Robots

In this section, we discuss our applications with respect to the functionalities introduced in this chapter. In particular, we describe the robots on which we implemented an online semantic mapping mechanism, their equipment and tasks. We evaluate our system by first analyzing the errors introduced by each of its components, and then by measuring the overall performance of the robot. In our thought experiment (Section 6.3.1) we considered a semantic mapping system that was not influenced by "external" factors. Conversely, we are now interested in understanding how much a semantic-driven system is affected by very simple, non-sophisticated components – low-cost solutions – in terms of behavior execution and task efficacy. In particular, we set up a quasi-worst-case experiment, in which we explicitly evaluate the behavior of the robot in a difficult scenario.

Platforms, Sensors and Functionalities For implementing our prototype system, we considered multiple mobile robots and we developed our code in a platform independent way, by means of ROS[4], C++, Python and Prolog. In particular, we implemented all our algorithms for learning and executing situated behaviors in C++, while using Python for the deliberative part of our system. This part directly interfaces with a planner and a knowledge base implemented in Prolog. Our code

[4]Robot Operating System, `http://www.ros.org`



(a) DIAGO, a mobile robot derived from a Segway platform.

(b) Turtlebot interacting with human during online semantic mapping.

Figure 6.7. Examples of deployment platforms for our prototype system of service robot.

was deployed on DIAGO – a modified version of a Segway platform (Figure 6.7a), as well as on Videre Erratic, a Turtlebot (Figure 6.7b), a student-built robot named MARRtino and, more recently, on a YouBot. Each robot was endowed with an RGB-D camera, a laser rangefinder, a microphone and one or more speakers. In addition to the sensory arrangement, each robot was equipped with a navigation stack (from ROS), a natural language processing chain based on frames (for interpreting commands), as well as a vision system. This system is capable of (1) recognizing the light of a laser pointer in the environment, (2) segmenting objects, (3) recognizing objects through the use of SURF features. The light from the laser pointer can be used by a human, together with the speech recognition, to generate an event for the online semantic mapping algorithm, as in Figure 6.7b. To approximate spatial information and generate predicates $\mathcal{P}_s$ when new information is stored in the semantic map, we use the A-Grid [20]. Despite the structural bias introduced by such approximation (see Section 6.1.2), we are interested in evaluating the behaviors of the robots according to the learned knowledge, rather than their actual precision in terms of centimeters.

Scenarios and Tasks We deploy our robots in two different domestic and office scenarios, with the goal of interacting with people to acquire knowledge about the environment. Our robots typically start with an empty knowledge base, or from the state in which their execution was interrupted at previous times. Then, the knowledge base is filled to learn semantic information and improve the execution of

complex and interactive behaviors. To this end, our robots all implement the same online semantic mapping algorithm, and continually acquire semantic information through it. Occasionally, they also observe demonstrations of specific tasks (by means of tele-operation). These tasks may include following/approaching a person, approaching an object, etc. Demonstrations are collected in a dataset that is used by the robot to learn affordance models that are greedily maximized to generate an affordance driven situated policy. Users can instruct robots to bring objects in some places, search for something, and more complex tasks that are just sequencing of simple behaviors for a mobile robot.

Component Evaluation: Situated Behaviors For the purposes of our experiment, we consider a library composed by one single situated behavior – i.e., approaching a generic object given the current state. In fact, while we show a more effective use of spatio-temporal affordances in Chapter 7, in this section we aim at evaluating the effects of bad underlying functionalities on high-level behaviors generated through the mechanism of online semantic mapping. For this reason, we train an affordance model, for this task, by means of only 3 demonstrations provided by a non-expert user. The affordance is greedily used for generating a policy that reproduces demonstrated behaviors with an (as expected high) average error of 9 cm for the distance and 22 degrees for the angle.

Component Evaluation: A-Grid As already discussed above in this chapter, the A-Grid is an efficient structure for storing spatial information, and for discretizing 2D metric maps. It consists of a non-uniform grid whose cells vary as a function of the occupied areas of the map, and whose compression rate with respect to the original metric map is $\sim 98.5\%$ [43]. However, its average structural bias is estimated as $57\% \pm 18.5$ with respect to the real size of the represented spatial element. Due to this, an object memorized in $\mathcal{SM}$ through the A-Grid is represented, on average, almost 1.5 times bigger than its real size.

Component Evaluation: Vision System For our robots, we consider a very simple and low-cost vision system that performs depth and color based segmentation. The approach, in particular, presents an average Detection Rate of ~ 0.8, and an average False Alarm Rate of ~ 0.13. In addition, the size of the objects extracted by means of our segmentation algorithm typically presents a mean percentage error of ~ 0.25.

Component Evaluation: Speech Recognition The effectiveness of the speech component is measured with respect to the quality of both the transcriptions of the user utterances, and the command interpretation. The former are evaluated in terms of Word Error Rate, scoring a value of ~ 0.26. The second, instead, is measured in terms of precision, recall and F1-measure. In particular, commands are recognized with a precision of ~ 75.5, a recall of ~ 68.0 and an F1 of ~ 71.5.

Integrated System Evaluation Given these high-error, simple components, our goal is to evaluate how the whole system is affected during a typical task executed

in a real environment. For this reason, we deployed our application by implementing the online semantic mapping algorithm both in an office and domestic scenario. We asked 6 expert and 10 non-expert users to drive the robot around the environment, by means of the vocal interface, and provide commands to the agent. We also asked them to generate events for the robot, through a laser pointer and the vocal interface. In this application, such events are used to insert and update an object in the semantic map. As explained before, events are triggered by two components: a laser pointer – used to highlight the object – and a spoken dialogue – used to retrieve the semantic label of the object. The robot started with no knowledge about the environment and, initially, no command was properly executed, if related to objects or locations in the environment. After having memorized different objects, locations and properties in the robot knowledge base, the users were asked to

1. order the robot to move in front an object (5 times for each object);

2. count the number of times the robot was successful in executing its task.

Although almost 50% of the objects were placed with an error greater than 30cm, only 7% of the times the users considered the task as failed at execution time. Hence, despite the approximations that have been introduced in the representation, the robot is still able to execute the task in a satisfactory way for the human. This confirms our intuitions from the thought experiment – i.e., that the performance of the robot increases when more knowledge is available to it.

6.4. Contributions

In this chapter, we discussed the generation of robot behaviors in the context of deliberative systems that make use of semantic maps. In particular, we introduced a novel approach for gathering semantic information (Section 6.1) and generating adaptive behaviors while collecting new knowledge about the world (Section 6.2). To evaluate our method without the error chaining induced by perception and planning functionalities, we presented a thought experiment. Such experiment confirmed our insights about the importance of semantic reasoning for increased autonomy and adaptation capabilities. Then, we described our implementation, and we extended our thought experiment to a sort of worst-case scenario. In this way, we have been able to study the applicability of our solution in difficult domains, where perception and actuation routines might fail due to external reasons.

The main contributions of this chapter are represented by (1) the definition of a principled methodology for collecting semantic information by using our model of semantic maps; (2) the definition, and task-oriented evaluation, of a procedure for using the acquired knowledge to generate behaviors via reasoning and inference. In particular, this chapter extends our prior work [9, 43] by extending the online semantic mapping algorithm to the general case of a representation that is coherent with the model of Chapter 4.

Chapter 7

Policy Learning With Spatio-Temporal Affordances

In the behavior of all men, and particularly of rulers, against whom there is no recourse at law, people judge by the outcome.

— Niccolò Machiavelli

In this Chapter, we move our focus from deliberation to situated action, where we adopt Markov decision processes to generate robot policies and simultaneously learn spatio-temporal affordances, as defined in Chapter 5. In particular, in this chapter, we introduce our approach for learning action policies through Monte Carlo tree search and data aggregation. While collecting cost-to-go information relative to the task, in each visited state we enable our robots to concurrently acquire a model of affordances. Then, at every time t, spatio-temporal affordances are adopted to guide the Monte Carlo exploration and reduce the search space. Coherently with our statement for this thesis, we adopt the tools of policy learning through the MDP setting to generate situated behaviors, that are driven by the action semantics implicitly encoded within spatio-temporal affordances.

This chapter is organized as follows. First, we introduce our assumptions and problem formulation, as well as a general procedure for learning action policies through Monte Carlo tree search and data aggregation (Section 7.1). Then, we present our algorithm for learning situated robot policies through the use of spatio-temporal affordance maps (Section 7.2). The approach is experimentally validated in Section 7.3. Finally, we further discuss our work in terms of future directions and extensions to the approach (Section 7.4).

Assumptions and Problem Formulation

In this chapter, we formalize our learning problem by adopting the Markov Decision Process notation, as introduced in Section 2.14. In particular, we consider a category of problems in which $\mathcal{S}$ represents the state space of the environment, $\mathcal{A}$ consists of a *discrete* set of actions and $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function (or system dynamics). Throughout this chapter we also assume to access the dynamics

of the environment only through samples obtained by executing a policy in the real world or in a simulator. In our learning setting, we observe an *immediate reward* $R(s)$ in state $s \in \mathcal{S}$, bounded in $[0, 1]$. In addition to this, in some occasions we also observe demonstrations of a sub-optimal policy by means of state-action pairs collected in a dataset $\mathcal{D}_0$.

7.1. Monte Carlo Search With Data Aggregation

In this section, we introduce a method for policy improvement with data aggregation [123, 122, 22] and Monte Carlo tree search, that we leverage in the next section to learn semantic-driven situated behaviors through spatio-temporal affordances. This method – Monte Carlo Search with Data Aggregation (MCSDA) – is an iterative algorithm that, at each iteration $i \in \{0, ..., N\}$, generates a new policy π_i that improves π_{i-1} through short Monte Carlo roll-outs. In particular, such improvements are obtained by directly executing π_{i-1} and aggregating the rewards measured over several Monte Carlo simulations to the rewards at earlier iterations. As previous work [123, 122, 22], MCSDA leverages a supervised learning approach to learn a policy by means of a classifier.[1]

In its simplest form the algorithm takes as input a set $\mathcal{D}_0$ of state-action pairs (e.g., obtained from expert demonstrations) and proceeds as follows. First, MCSDA learns a classifier π_0 by using $\mathcal{D}_0$. Then, during each iteration i, the algorithm extends its dataset $\mathcal{D}$ by:

1. executing the previous policy π_{i-1} and generating a state s_t at each time-step t;

2. selecting for each s_t an action a_t that maximizes the expected value $Q(s_t, a)$ (Eq. (2.3)) of performing action the a in s_t. $Q(s_t, a)$ is estimated through Monte Carlo tree search;

3. aggregating the new state-action pairs – at each time-step t – to the previous dataset $\mathcal{D}$.

Finally, the aggregated dataset is used to train a new classifier π_i that substitutes the policy used at the previous iteration. The details of MCSDA are provided in Algorithm 6.

By relying on data aggregation, MCSDA generates a sequence $\pi_1, \pi_2, \dots \pi_N$ of policies and preserves the main characteristics of algorithms like AggreVate [122] – i.e., (1) it builds its dataset by exploring the states that the policy will probably encounter during its execution, (2) it can be interpreted as a Follow-The-Leader algorithm that learns a good classifier over all previous data and (3) can be easily transformed to use an online learner by simply using the dataset in sequence. However, the implementation of MCSDA is more practical due to the reduced amounts of roll-outs generated from the Monte Carlo simulation.

[1] Note that, since the chosen actions influence the distribution of states, our supervised learning problem is characterized by a non-i.i.d. dataset

Input: $\mathcal{D}_0$: dataset of initial state action pairs $\{s, a\}$
Output: π_N: policy learned after N iterations of the algorithm.
Data: $\mathcal{A}$: discrete set of actions; N: number of iterations of the algorithm; Δ: initial state distribution; K: number of Monte Carlo simulations; H: simulation steps.

```
begin
    // Learn initial policy
    Train classifier π_0 on D_0.
    Initialize D ← D_0.
    // Iterate the algorithm
    for i = 1 to N do
        // Sample state from initial distribution
        s_0 ← random state from Δ.
        // Iterate over the time-steps
        for t = 1 to T do
            // Execute the policy in the state at time t − 1 and generate
               a new state
            Get state s_t by executing π_{i−1}(s_{t−1}).
            A ← select or sub-sample (if needed) feasible actions in s_t.
            // Iterate over actions
            foreach a ∈ A do
                // Estimate the value of executing a in state s_t
                Run K Monte Carlo simulations of length H to estimate Q(s_t, a).
            end
            // Choose the action that maximizes the estimated value
            a_t ← arg max_a Q(s_t, a)
            // Insert in the dataset the obtained state s_t and the best
               action
            D ← D ∪ {s_t, a_t}
        end
        // Learn a new policy
        Train classifier π_i on D.
    end
    return π̃_N
end
```

Algorithm 6: Monte Carlo Search with Data Aggregation.

7.2. Policy Improvement with Spatio-Temporal Affordances

In this section, we present Policy Improvement with Spatio-Temporal Affordance Maps (π-STAM), an iterative method for learning spatial affordances and generating robot behaviors. In particular, π-STAM is a model-based reinforcement learning algorithm that builds on MCSDA and, additionally, uses aggregated datasets to improve spatio-temporal affordance models. Our goal consists in generating a policy that is composed of a discrete set of actions and adapts to the unknown action semantics of the environment. As in the schema proposed in Section 7.1, π-STAM uses a classifier to generate a policy which is continuously refined through the aggregation [123] of an initial dataset with roll-outs collected through Monte Carlo

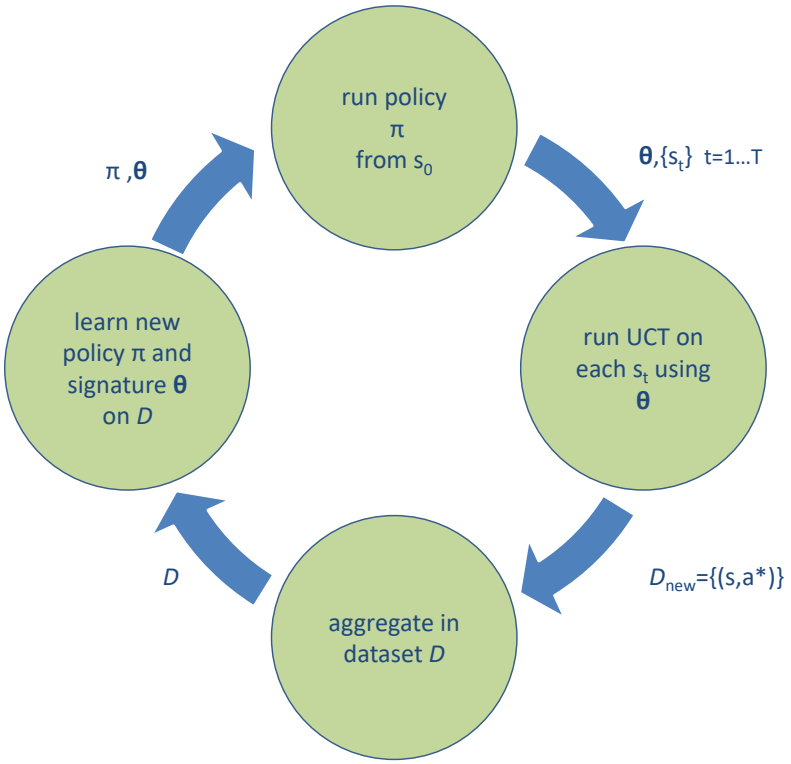


Figure 7.1. π-STAM with time-invariant affordance signatures.

tree search [15]. While generating action policies, the same process is also used to improve spatio-temporal affordance models. These are then used to evaluate promising actions, and to reduce the search space of the Monte Carlo exploration. Using this mechanism, situated behaviors are constructed according to the semantic information implicitly modeled by the affordances.

More specifically, the goal of π-STAM is to generate, at each iteration $i \in \{1, ..., N\}$, a new policy π_i that improves π_{i-1}, given an initial π_0. To illustrate π-STAM, in Section 7.2.1 we first consider time-invariant (Remark 5.5) affordance signatures and, then, we extend the algorithm to the general, time-dependent case (Section 7.2.2). Additionally, we adapt the notation of the STA function (Definition 5.1) from $f_{E,\mathcal{T}}$ to $f_{E,\mathcal{A}}$, thus explicating our intention to model the affordance function f_{E,a_i} of each action $a \in \mathcal{A}$.

7.2.1. Time-invariant Affordance Signatures

In the case of time-invariant affordances, π-STAM evolves as described in Algorithm 7. The algorithm takes as input a dataset $\mathcal{D}_0$ of state-action pairs to generate both an initial policy π_0 and the affordance signature[2] $\boldsymbol{\theta}_0$. As in Section 7.1, a policy π is generated through a dataset $\mathcal{D}$ by using a supervised learning approach – a generic classifier. Then, during each iteration, the algorithm proceeds as follows (Figure 7.1):

[2]For Algorithm 7 we refer to affordance signatures as $\boldsymbol{\theta}_i$, where i is the iteration of the algorithm.

1. it executes the previous policy π_{i-1} for T time-steps and generates a set of T states $\{s_t \mid t = 1 \dots T\}$;

2. it runs, for each s_t, the Upper Confidence Bound for Trees (UCT) algorithm [15]. As introduced in Section 7.1, UCT is an iterative algorithm that, at each iteration $h = 1 \dots H$, simulates the execution of each legal action in $s_{t+(h-1)}$ and selects the best action a^*_h as

$$e = C \cdot \sqrt{\frac{\log(\sum_a n(s_{t+(h-1)}, a))}{n(s_{t+(h-1)}, a)}} \tag{7.1}$$

$$a^*_h = \arg\max_a Q_i(s_{t+(h-1)}, a) + e, \tag{7.2}$$

   where $Q_i(s_{t+(h-1)}, a)$ is the action-value function (Eq. (2.3)) of action a in state $s_{t+(h-1)}$, C is a constant that multiplies and controls the exploration term e, and $n(s_{t+(h-1)}, a)$ is the number of occurrences of a in $s_{t+(h-1)}$. The value function $Q_i(s, a)$ is obtained by back-propagating the final reward of each simulation to all the traversed states s, according to the chosen action a. After selecting the best action a^*_h, a new state s_{t+h} is generated by the transition function T of the MDP, and all the obtained state-action pairs $\{(s_{t+(h-1)}, a^*_h) \mid h = 1 \dots H\}$ are collected in a dataset $\mathcal{D}_{new}$. Legal actions in $s_{t+(h-1)}$ are selected from $\mathcal{A}$ according to the STA function $f_{E,\mathcal{A}}(s_{t+(h-1)}, \boldsymbol{\theta}_{i-1})$. In particular, actions are considered to be legal if their likelihood to be afforded in $s_{t+(h-1)}$ is higher than a threshold ρ that is adaptively determined according to

$$\rho = \frac{1}{2} f_{max} = \frac{1}{2} \max_{a \in \mathcal{A}} f_{E,a}(s_{t+(h-1)}, \boldsymbol{\theta}_{i-1}), \tag{7.3}$$

   where f_{max} is the maximum affordance value with respect to the actions $a \in \mathcal{A}$. Additionally, actions whose likelihood is lower than ρ can be randomly selected to be legal with ϵ probability. This is generally useful to avoid over-fitting to wrong affordance models, especially during the first iterations of the algorithm.

3. it aggregates the new state-action pairs contained in $\mathcal{D}_{new}$ to the previous dataset $\mathcal{D}$. Note that: (1) the dataset $\mathcal{D}$ is non-i.i.d., since chosen actions influence the distribution of states; (2) while aggregating new state-action pairs, previously seen states are removed from the original dataset to avoid duplicates with different action labels and to allow information revision in the algorithm;

4. it uses the aggregated dataset $\mathcal{D}$ to train a new classifier π_i that substitutes the policy used at the previous iteration, as well as the new signature $\boldsymbol{\theta}_i$ of the STA function. Such signature is chosen to maximize Eq. (5.4) – here reported for clarity:

$$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta}_j} \sum_{\{s:(s,a_j) \in \mathcal{D}\}} \log f_{E,a_j}(s, \boldsymbol{\theta}_j), \tag{7.4}$$

Input: $\mathcal{D}_0$: dataset of initial state action pairs $\{s, a\}$
Output: π_N: policy learned after N iterations of the algorithm; $\boldsymbol{\theta}_N$: signature of the time-invariant STA function, learned after N iterations of π-STAM.
Data: $\mathcal{A}$: discrete set of actions; N: number of iterations of the algorithm; Δ: initial state distribution; H: UCT horizon; T: policy execution time-steps; $\mathcal{D}$: aggregated dataset.

begin
  `// Learn initial policy`
  Train classifier π_0 on $\mathcal{D}_0$.
  `// Initialize affordance signature for each action`
  for $j = 1 \dots |\mathcal{A}|$ **do**
    Initialize $\boldsymbol{\theta}_{j,0} = \arg\max_{\boldsymbol{\theta}_j} \sum_{\{s:(s,a_j)\in\mathcal{D}_0\}} \log f_{E,a_j}(s, \boldsymbol{\theta}_j)$
  end
  Initialize $\mathcal{D} \leftarrow \mathcal{D}_0$.
  `// Iterate the algorithm`
  for $i = 1$ **to** N **do**
    `// Sample state from initial distribution`
    $s_0 \leftarrow$ random state from Δ.
    `// Iterate over time-steps`
    for $t = 1$ **to** T **do**
      `// Execute the policy in the state at time` $t-1$ `and generate a new state`
1)   Get state s_t by executing $\pi_{i-1}(s_{t-1})$.
      `// Generate good state-action pairs according to their value estimated by UCT with spatio-temporal affordances`
2) $\mathcal{D}_{new} \leftarrow \text{UCT}(\mathcal{A}, s_t)$.
      `// Aggregate the dataset`
3) $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{new}$.
    end
    `// Learn a new policy`
4) Train classifier π_i on $\mathcal{D}$.
    `// Generate a new affordance signature`
    for $j = 1 \dots |\mathcal{A}|$ **do**
      $\boldsymbol{\theta}_{j,i} = \arg\max_{\boldsymbol{\theta}_j} \sum_{\{s:(s,a_j)\in\mathcal{D}\}} \log f_{E,a_j}(s, \boldsymbol{\theta}_j)$.
    end
  end
  return π_N, $\boldsymbol{\theta}_N$
end

Algorithm 7: π-STAM with time-invariant affordance signatures.

where $j = 1 \dots |\mathcal{A}|$. As discussed in Chapter 5, this optimization consists of finding parameters $\boldsymbol{\theta}_{j,i}$, at iteration i, by maximizing the likelihood of the portion of the dataset labeled with the considered action a_j.

The intuition behind this algorithm is the following. On the one side, given a good STA model and a state s, the (legal) actions that UCT needs to explore in s can be directly chosen according to their affordability. On the other side, if an action a in state s is estimated to be of high value for the policy π, the affordance value of a should be increased. To increase the affordance value of a in s, the pair (s, a) should be included in the dataset over which the signatures are learned.

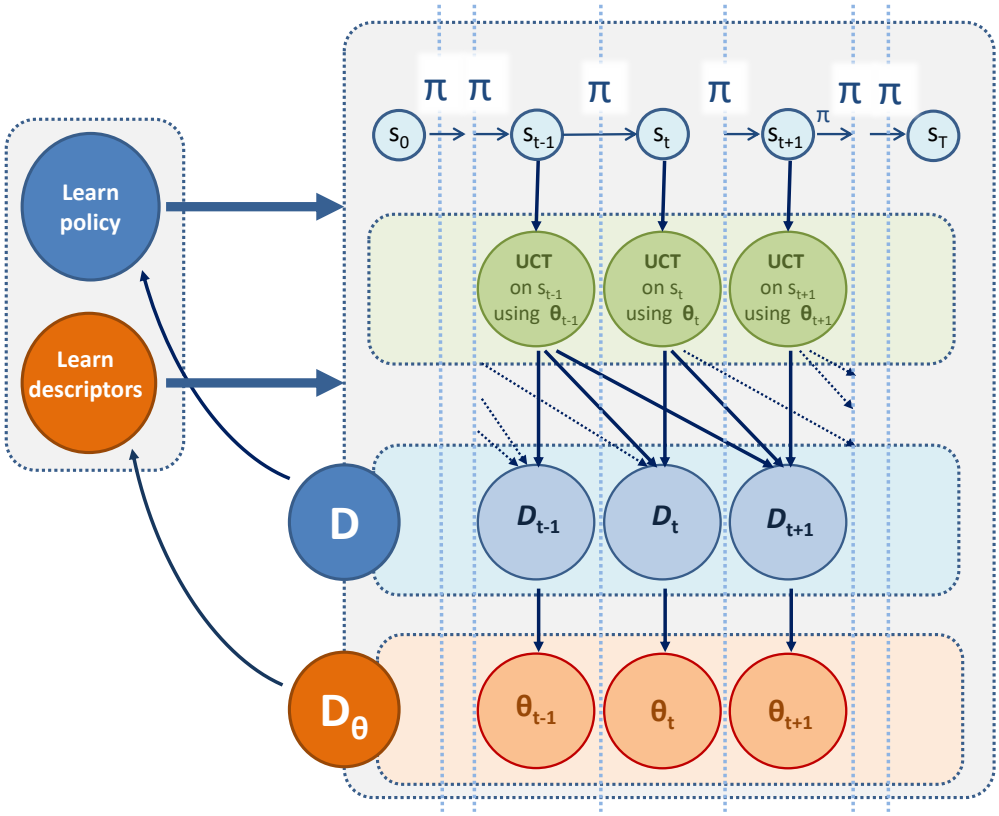


Figure 7.2. π-STAM with time-dependent affordance signatures.

In the next section, we illustrate a similar mechanism for the case of time-dependent affordance signatures.

7.2.2. Time-dependent Affordance Signatures

In this section, we refer to affordance signatures as $\boldsymbol{\theta}_{t,i}$ and to affordance descriptors as $\boldsymbol{\lambda}_i$, where t is the considered time-step, and i is the iteration of the algorithm. In the case of time-dependent affordance signatures, π-STAM takes as input a dataset $\mathcal{D}_0$ of state-action pairs, generates an initial policy π_0, and evolves as described in Algorithm 8. In particular, at each iteration, the algorithm proceeds as follows (Figure 7.2):

1. it executes the previous policy π_{i-1} for T time-steps and, at each time-step t:
   (a) it generates a state s_t by means of the policy;
   (b) it executes, starting from s_t, the UCT algorithm for $h = 1 \dots H$ iterations and it generates a dataset of state-action pairs $\mathcal{D}_{new} = \{(s_{t+(h-1)}, a^*_h) \mid h = 1 \dots H\}$. Legal actions in state $s_{t+(h-1)}$ are selected from $\mathcal{A}$ according to the STA function $f_{E,\mathcal{A}}(s_{t+(h-1)}, \boldsymbol{\theta}_{t+(h-1),i-1})$. In particular, actions are considered to be legal if their likelihood to be afforded in $s_{t+(h-1)}$ is

Input: $\mathcal{D}_0$: dataset of initial state action pairs $\{s, a\}$
Output: π_N: policy learned after N iterations of the algorithm; $\boldsymbol{\lambda}_N$: descriptors of the time-dependent STA function, learned after N iterations of π-STAM.
Data: $\mathcal{A}$: discrete set of actions; N: number of iterations of the algorithm; Δ: initial state distribution; H: UCT horizon; T: policy execution time-steps $\mathcal{D}$: aggregated dataset; $\mathcal{D}_{\boldsymbol{\theta}}$: dataset of parameters $\boldsymbol{\theta}$.

begin
  `// Learn initial policy`
  Train classifier π_0 on $\mathcal{D}_0$.
  `// Initialize affordance signature randomly for each action`
  for $t = 1$ **to** T **do**
    Initialize $\boldsymbol{\theta}_{j,t,0}$ as random.
  end
  Initialize $\mathcal{D} \leftarrow \mathcal{D}_0$.
  for $i = 1$ **to** N **do**
    `// Sample from initial distribution`
    $s_0 \leftarrow$ random state from Δ.
    `// Initialize dataset for learning descriptors from signatures`
    $\mathcal{D}_{\boldsymbol{\theta}} \leftarrow \emptyset$.
    for $t = 1$ **to** T **do**
      `// Execute policy in state at` $t-1$ `and generate new state`
      Get state s_t by executing $\pi_{i-1}(s_{t-1})$.
      `// Generate good state-action pairs according to their value estimated by UCT with spatio-temporal affordances`
      $\mathcal{D}_{new} \leftarrow \mathrm{UCT}(\mathcal{A}, s_t)$
      for $h = 1 \dots H$ **do**
        `// Extract state-action pairs for time-step` $t + (h-1)$
        $\{(s_{t+(h-1)}, a_h^*)\} \leftarrow \mathrm{extract}(\mathcal{D}_{new}, h)$
        `// Aggregate extracted data to the corresponding dataset`
        $\mathcal{D}_{t+(h-1)} \leftarrow \mathcal{D}_{t+(h-1)} \cup (s_{t+(h-1)}, a_h^*)$
      end
      `// Generate signature of time` t `by using dataset` $\mathcal{D}_t$
      for $j = 1 \dots |\mathcal{A}|$ **do**
        $\boldsymbol{\theta}_{j,t,i} = \arg\max_{\boldsymbol{\theta}_j} \sum_{\{s:(s,a_j)\in\mathcal{D}_t\}} \log f_{E,a_j}(s, \boldsymbol{\theta}_j)$.
      end
      `// Extract states in` $\mathcal{D}_t$
      $\mathcal{S}_{\mathcal{D}_t} \leftarrow \mathrm{getStates}(\mathcal{D}_t)$
      `// Store extracted states with the signature of time` t
      $\mathcal{D}_{\boldsymbol{\theta}} \leftarrow \mathcal{D}_{\boldsymbol{\theta}} \cup \mathrm{labelData}(\mathcal{S}_{\mathcal{D}_t}, \boldsymbol{\theta}_{t,i})$
      `// Aggregate dataset`
      $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{new}$
    end
    Train classifier π_i on $\mathcal{D}$.
    $\boldsymbol{\lambda}_i \leftarrow$ Parameters from regressor trained on $\mathcal{D}_{\boldsymbol{\theta}}$
  end
  return π_N, $\boldsymbol{\lambda}_N$
end

Algorithm 8: π-STAM with time-dependent affordance signatures.

higher than a threshold ρ, that is adaptively determined according to

$$\rho = \frac{1}{2} \max_{a \in \mathcal{A}} f_{E,a}(s_{t+(h-1)}, \boldsymbol{\theta}_{t+(h-1),i-1}). \tag{7.5}$$

As in Algorithm 7, actions whose likelihood is lower than ρ can be randomly selected to be legal with ϵ probability;

(c) it aggregates each element of the new set $\{(s_{t+(h-1)}, a^*_h)\}$ to the appropriate[3] dataset $\mathcal{D}_{t+(h-1)}$;

(d) it aggregates the $\mathcal{D}_{new}$ to the dataset $\mathcal{D}$;

(e) it uses $\mathcal{D}_t$ to train a signature $\boldsymbol{\theta}_{t,i}$ as in Eq. (5.3) – here reported for clarity:

$$\boldsymbol{\theta}_{t,j} = \arg\max_{\boldsymbol{\theta}_{t,j}} \sum_{\{s_t:(s_t,\tau_j)\in\mathcal{D}_t\}} \log f_{E,\tau_j}(s_t, \boldsymbol{\theta}_{t,j}). \tag{7.6}$$

$\boldsymbol{\theta}_{t,i}$ is then stored in a dataset $\mathcal{D}_{\boldsymbol{\theta}}$;

2. it uses the aggregated dataset $\mathcal{D}$ to train a new classifier π_i that substitutes the policy used at the previous iteration;

3. it uses $\mathcal{D}_{\boldsymbol{\theta}}$ to learn the descriptors $\boldsymbol{\lambda}_i$ as the parameters of a regression function.

Differently from its simpler version, π-STAM with time-dependent affordance signatures aggregates multiple datasets. In fact, in addition to $\mathcal{D}$ – that is used to generate the policy π, the algorithm stores, for each $t = 1 \dots T$, an aggregated dataset $\mathcal{D}_t$ that is used to train the signature $\boldsymbol{\theta}_t$. All the signatures are then collected in $\mathcal{D}_{\boldsymbol{\theta}}$, that is used for learning a regression function whose parameters are the affordance descriptors $\boldsymbol{\lambda}$.

7.2.3. Semantic-driven Situated Behaviors

π-STAM is a method that leverages Monte Carlo tree search to generate robot policies by means of spatio-temporal affordance maps. In Chapter 5, we characterized spatio-temporal affordance maps as task-directed representations, that describe a given environment in terms of its action semantics. Additionally, we explained that spatio-temporal affordances implicitly encode the semantics of an action by expressing its "goodness", given a state of the world. Following these observations, in this section we argue that policies generated through our algorithm are semantic-driven situated behaviors. In fact, during each iteration of π-STAM, the search space of the Monte Carlo search is restricted by the affordance model that is continuously refined. Hence, policy improvement is restricted to the action space induced by the affordance model. Since affordances encode action semantics, a policy obtained in the search space induced by a STA is a semantic-driven policy.

Remark 7.1. *A policy obtained through π-STAM is a semantic-driven situated behavior, since it is generated through affordances that encode action semantics.*

[3]Note that, in Algorithm 8, the aggregation can be carried out directly at time t, since iterations of UCT starting at $t + 1$ affect only datasets of future times.

Figure 7.3. RoboCup scenario for the evaluation of MCSDA.

7.3. Experimental Validation

In this section, we validate our approach both on a real NAO platform (V4) and on simulations running on a single Intel Core i7-5700HQ core, with CPU@2.70GHz and 16GB of RAM. In particular, we first measure the effectiveness of MCSDA in learning a robot policy (Section 7.3.1). Then, we evaluate the impact of affordances both by comparing π-STAM against MCSDA (Section 7.3.2) and by qualitatively evaluating learned affordance models according to the state of the world.

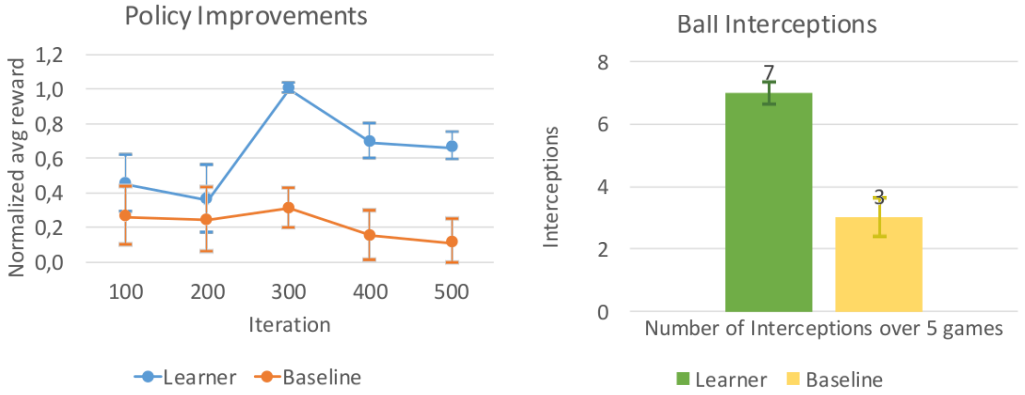
7.3.1. Monte Carlo Search with Data Aggregation

In this section, we consider the setup proposed by the RoboCup Standard Platform League (SPL), where NAO robots compete in a 5-vs-5 soccer game (Figure 7.3). Our goal consists in generating a robot defender policy that adapts to the strategy of the opponent team. Such strategy is not known and, hence, the world dynamics is partially observable and difficult to model. In fact, RoboCup is a dynamic adversarial environment where robots must adapt to the surrounding world quickly and efficiently. For this reason, we evaluate our learning approach in the short range after few number of simulation steps.

MCSDA with NAO and RoboCup To apply MCSDA to the RoboCup context, we represent the state of our problem as a tuple (p_r, p_b, v_b), where $p_r = (x_r, y_r)$ is the position of the robot, $p_b = (x_b, y_b)$ the position of the ball in the field, and $v_b = (v_{xb}, v_{yb})$ its velocity. Additionally, we consider a discrete set of actions composed by: `stand` (the robot does not move), `move_up` (the robot moves forwards), `move_down` (the robot moves backwards), `move_left`, `move_right`. The reward function is

$$R(s) = \frac{\texttt{MAX\_FIELD\_DISTANCE} - |p_r - p_b|}{\texttt{MAX\_FIELD\_DISTANCE}}, \tag{7.7}$$

where `MAX_FIELD_DISTANCE` corresponds to the game-field diagonal. Given this reduced domain representation, we use MCSDA with expert demonstrations collected from the opponent team. We run Monte Carlo simulations both on a simplified



(a) Normalized average reward of the learner and baseline after different MCSDA iterations.

(b) Sum of intercepted ball over five matches after different MCSDA iterations.

Figure 7.4. Evaluation of MCSDA after different iterations.

simulator and the C++ B-Human simulator[4], and we set the roll-out horizon $H = 3$. This value has been found to be a good trade-off between in-game performance improvement and usability of the approach. Extending the horizon, in fact, improves the player performance at the cost of more computational resources.

Results The goals of our learner consist in (1) improving defender's performance while playing against opponent robots and (2) decreasing the number of opponent scores while intercepting as many balls as possible. Here, we analyze the average reward (Figure 7.4a) of our agent as well as the number of summed ball interceptions (Figure 7.4b) and the final score (Table 7.1) obtained during five regular games, after $100, 200, 300, 400, 500$ iterations of the algorithm. To quantify the improvement of the learned behavior, we compare MCSDA against the fixed defender policy used for initializing the algorithm. In particular, for our evaluation we use two teams, one featuring a learning defender, the other with the fixed policy defender. Figure 7.4a shows that MCSDA always learns a better policy with respect to the baseline, with a drop in performance between game 3 and 4 that is probably due to game factors, such as player penalization and ball positioning rules. Such drop only has a marginal impact with respect to previous improvements. Figure 7.4b, instead, shows the sum of intercepted balls of the two teams on the same set of games as before. The increased number of interceptions is confirmed by the final

[4] `https://www.b-human.de/`

Table 7.1. Final scores of five matches after different MCSDA iterations.

	MCSDA iterations				
Teams	100	200	300	400	500
Learning	2	3	0	1	1
Non-learning	3	2	1	1	1

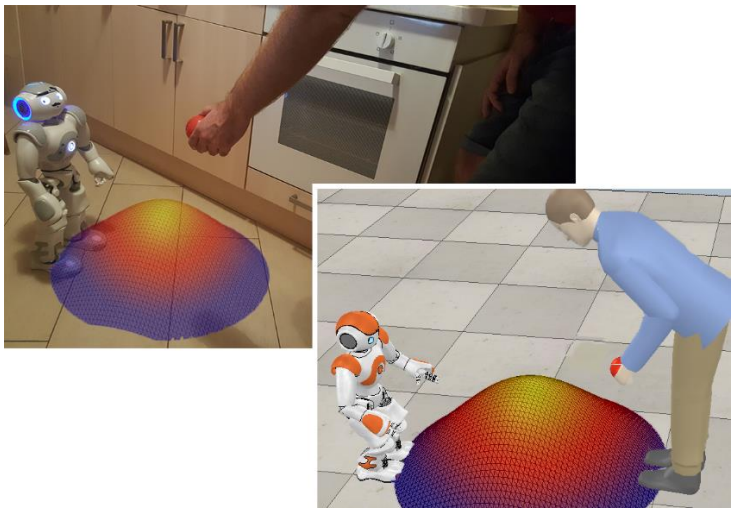


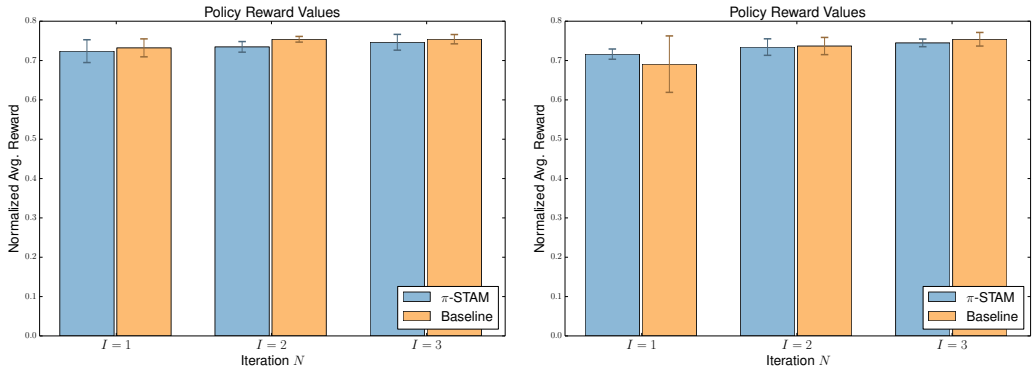
Figure 7.5. Human-robot handover with π-STAM both in real world and simulation.

scores reported in Table 7.1, where the score of the opponent team decreases as the learner refines its policy. Since we learn the policy of a defender, we use as a metric the number of received goals rather than the number of scored goals.

7.3.2. Policy Improvement with Spatio-Temporal Affordance Maps

Human-robot handovers are difficult tasks, where the large state space of the problem greatly affects the usability of policy learning methods. Since the goal of π-STAM is to refine an affordance model that reduces the search space of the policy, in this section we evaluate our algorithm on this problem – after a small number iterations. In particular, we test the effectiveness of π-STAM to obtain a good handover policy and yet to reduce (1) the number of simulated action executions in UCT and (2) the duration of an iteration of the algorithm. To this end, we compare against a baseline implemented through MCSDA. We evaluate the learned policies both on a V-REP simulated and real NAO robot by using the average reward obtained over 10 different trials. Additionally, we qualitatively evaluate the learned affordance models by observing their evolution as a function of the state of the world. Finally, we show an example of prior knowledge introduced in the affordance model. In particular, we embed in the model the "eye-contact" social rule, according to which no handover can be performed when the partner is not paying attention.

Human-robot Handovers with π-STAM In our experimental setup, the task of the humanoid robot is to take an object – a red ball – from the hands of a human operator. During the learning phase of the algorithm (1) the horizon is selected to be $H = 4$ to guarantee a good trade-off between performance improvement and usability of the approach, (2) the probability to randomly expand a non-legal action in UCT is set to $\epsilon = 0.3$, (3) the affordance function is implemented through Gaussian Mixture Models (GMMs) and (4) the policy is learned through GMM classification. Since



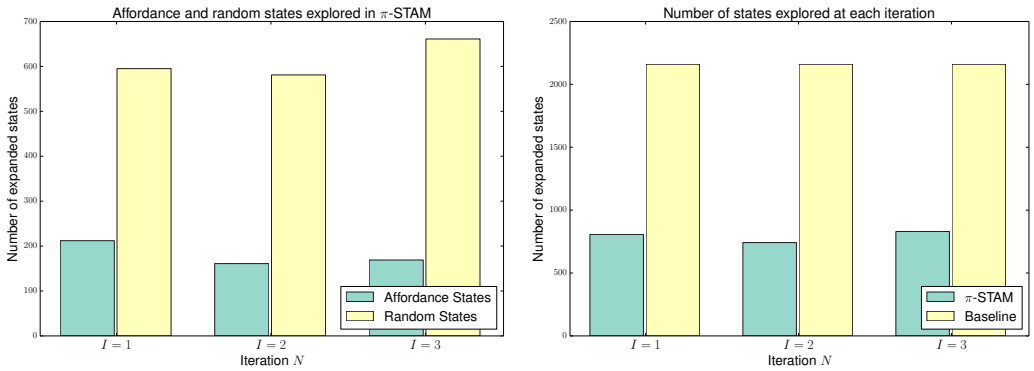
(a) Average reward obtained by π-STAM and the baseline in simulated experiments.

(b) Average reward obtained by π-STAM and the baseline in real world experiments.

Figure 7.6. Normalized average reward obtained by π-STAM and the baseline algorithm over 10 handovers, both in simulation and real world.

the GMMs are used for implementing the affordance function, the signature $\boldsymbol{\theta}$ of the STA is composed as a tuple $\boldsymbol{\theta} = \langle \pi_1, \mu_1, \Sigma_1, \ldots, \pi_N, \mu_N, \Sigma_N \rangle$, where π_i is the prior, μ_i the mean vector and Σ_i the covariance matrix of a mixture of N Gaussians. The state of the problem is composed of the Cartesian pose (position and angles) of the robot kinematic chains corresponding to the head and the two arms, together with the state of the hands (opened/closed). Additionally, the state includes the relative distance between the robot and object poses, the position of the target in the image frame of the cameras as well as an "attention bit" indicating if the human is looking towards the NAO. The robot is allowed to execute the following 27 actions: (1) rotate the head to the left, right, up and down; (2) move the body forward, backward, left, right and rotate it on the left and right; (3) move one of the two arms forward, backward, left, right, up and down; (4) close and open the hands; (5) execute the "null" action to stay still. The reward function $R(s)$, instead, is modeled to penalize the distance of the robot from the target, as well as orientations of the head where the object is not centered in one of the two cameras. Additionally, the reward penalizes the robot whenever its hands are closed before the object is reached.

Results The goal of our robot is to perform handovers and maximize the obtained reward. In our experiments, we analyze the average reward obtained by the agent, as well as its standard deviation, both in simulation and real world. Figure 7.6 shows the normalized average reward obtained by a NAO during 10 handovers, both in simulation (Figure 7.6a) and in real world (Figure 7.6b). In the figure, we compare the reward obtained by π-STAM and the baseline algorithm (MCSDA) over 3 iterations of the algorithms. Notice that, as shown in Figure 7.7, while the baseline fully explores all the actions during the execution of UCT, π-STAM only expands legal actions. Still, the difference in the obtained average reward only slightly favors the baseline algorithm and the generated policies perform similarly. Conversely, the computational load (Figure 7.7) and time consumption (Figure 7.8) of π-STAM are significantly reduced with respect to the baseline. In particular, the number of states



(a) Affordance and random expanded states in π-STAM.

(b) Total expanded states in π-STAM and the baseline.

Figure 7.7. Comparison between the mean number of states expanded by UTC in π-STAM and in the baseline algorithm. As shown in the top figure, in the former algorithm states are expanded as legal actions because of affordances or because they are randomly selected. In the latter case, instead, the reported value is constant because all actions are always evaluated.

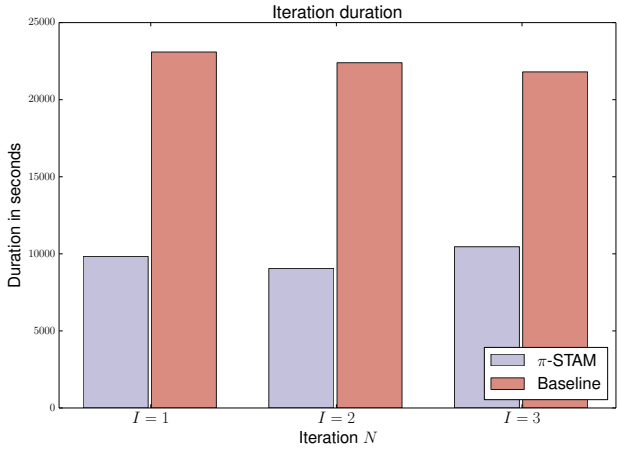
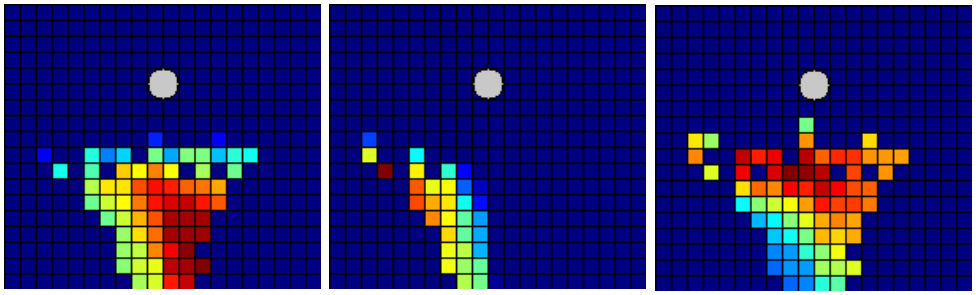


Figure 7.8. Comparison between the time consumption (in seconds) of 3 iterations of π-STAM and the baseline algorithm.

that π-STAM expands due to affordances (Figure 7.7a) is small. This demonstrates that π-STAM is able to efficiently capture the semantics of actions and, thanks to this, the search algorithm only spends time in "good" portions of the state space – those with higher expected reward. Notice that, as shown in Figure 7.7a, a significant part of the states expanded during the execution of π-STAM is randomly selected and depends on the chosen ϵ value. While this represents a lower-bound to the number of expanded states, it is essential to guarantee exploration in the affordance model and avoid overfitting to wrong state spaces during the first iterations of the algorithm. Although not discussed here in detail, it is possible to improve the efficiency of the algorithm by reducing the ϵ value when the number of iterations of

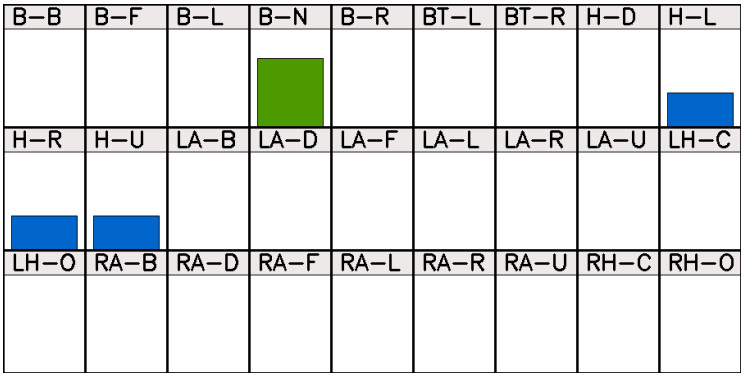


(a) Body forward **(b)** Head right **(c)** Head up

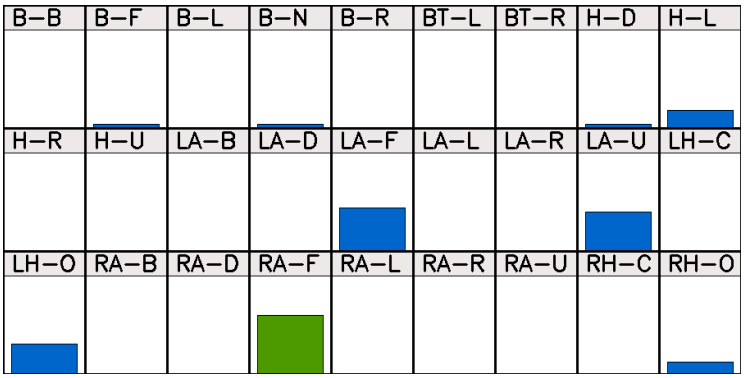
Figure 7.9. Heat-map of the spatial affordance distribution generated by π-STAM (after 3 iterations) for the body forward, head right and head up actions. The initial distribution was composed of states located at a distance between 45cm and 60cm in front of the target. Each cell of the grid has a size of approximately 5cm. Red colors represent high affordance values in the cell, while blue colors represent low values.

π-STAM increases. Intuitively, this enables the algorithm to first explore the state space and then to follow the learned affordance model proportionally to a confidence that increases over the number of iterations.

In addition to the learned policy, we qualitatively evaluate the obtained affordance model. Figure 7.9, for example, shows the heat-map – with a 5cm granularity – corresponding to the spatial affordance distribution of three actions: body forward (Figure 7.9a), head right (Figure 7.9b) and head left (Figure 7.9c). These have been generated from the model learned after 3 iterations of π-STAM with a small initial distribution composed of states located at a distance between 45cm and 60cm in front of a fixed target. From the figures, it can be observed that the learned affordances accurately model the need of the robot to (1) go forward when it is far from the target and stop at a distance of $\sim$20-25cm (from which the target is reachable with the arms), (2) turn the head right when it reaches positions on the left of the target, (3) increase the head pitch when it is close to the target (located at a position higher than the robot). Finally, Figure 7.10 shows the affordances of each action when implementing an "eye contact" social rule. Such rule states that the robot should wait for eye contact with the human partner to start the handover. In this test, we simply assume to have eye contact when the (Aldebaran) tracker of the NAO detects a face that is oriented towards the robot. As shown by results in Figure 7.10a, when the human is not paying attention, the highest affordance value (in green) corresponds to the "null" action. Additionally, head movements are allowed to search for eye contact. Conversely, when the partner looks at the NAO, the "right arm forward" action has the highest affordance value (see Figure 7.10b), and other arm-related actions are enabled. To provide this prior knowledge to the system, we initialize the signature $\boldsymbol{\theta}_0$ of the STA function with parameters of the GMM learned from a dataset where (1) all the actions are enabled when the "attention bit" is on and (2) only the head rotations and "null" actions are allowed when the "attention bit" is off.



(a) Learned affordance values for human-robot handovers in case of no eye-contact.



(b) Learned affordance values for human-robot handovers in case of eye-contact.

Figure 7.10. Affordance values for all the actions when the "eye contact" social rule is not respected and vice versa. The green bar corresponds to the maximum affordance action. Each action is named with the first letter corresponding to the considered kinematic chain (e.g. RA: right arm), while the second letter corresponds to the movement direction (e.g., B: backward, U: up, etc.). B-N, instead, corresponds to the "null" action.

7.4. Contributions

In this chapter we discussed the generation of situated robot policies by means of spatio-temporal affordances. In particular, we first introduced a general method for policy improvement based on Monte Carlo tree search (Section 7.1). Then, we extended this algorithm to use an affordance-driven Monte Carlo exploration and concurrently learn spatio-temporal affordances. In particular, we discussed π-STAM (Section 7.2) – Policy Improvement with Spatio-Temporal Affordance Maps, a novel practical algorithm that generates semantic-driven policies for autonomous agents through the combination of UCT and affordances. Even though the proposed approach can suitably learn action semantics, our algorithm still presents some limitations, such as (1) the use of expensive calls to a simulator and (2) the limited applicability to the large state-spaces of multi-agent systems, especially in the case of limited computational resources. While we attempt to relieve the first issue in the next chapter (Chapter 8), by learning dynamics models for robot control, we plan to extend this approach by using hierarchical affordance maps to further reduce the search space.

The main contributions of this chapter are represented by (1) the presentation and (2) extension of the MCSDA algorithm to learn a policy by means of spatio-temporal affordances; (2) the definition and evaluation of a novel procedure for learning both time-invariant and (3) time-dependent spatio-affordances in the context of MDPs. With respect to our prior work [117], this chapter extends π-STAM to the notion of time-dependent affordances introduced in Chapter 5, thus defining a full procedure for optimizing affordances and learning situated policies.

Chapter 8

Learning System Dynamics

On course doesn't mean perfect. On course means that even when things don't go perfectly, you are in the right direction.
— Charles Garfield

In Chapter 7 we extensively leverage simulations to access world dynamics and learn affordance-based situated behaviors. Unfortunately, simulator calls are generally costly and require high computational resources. For this reason, in this chapter we move our attention to approaches for learning accurate dynamics models, with limited access to physical system. This scenario is typical in robotics domains, where learning policies directly on robots is dangerous and non-trivial. To avoid this, in our work we develop a method for improving the multi-step prediction capabilities of the learned dynamics models. Of importance for this thesis, by using these improved models we can forward-simulate the dynamics of a system. Hence, we can directly use the learned models for generating situated behaviors within algorithms like MCSDA and π-STAM.

This chapter is organized as follows. First, we introduce the problem, the notation and the mathematical formulation (Section 8.1). Then, in Section 8.2, we describe our method and we experimentally validate it in Section 8.3.

8.1. Motivation and Problem Formulation

As introduced in Section 3.3, learning based approaches for controlling robots and autonomous agents are typically categorized into model-based [7, 55] and model-free [139, 85, 31, 72] methods. In this chapter, as in Chapter 7, we focus our attention on problems belonging to the former category, where a transition function – i.e., a dynamics model – is used for the creation of a control policy. In fact, as in the case of MCSDA and π-STAM, the generation of robot behaviors often requires the use of dynamics models that accurately capture the evolution of the environment. However, with the increasing complexity of both robotic technologies and application domains, analytically characterizing system dynamics is either difficult and error prone, or computationally expensive (as in the case of simulators). To tackle this problem, both physics-based methods [69] and black-box learning [6] have been

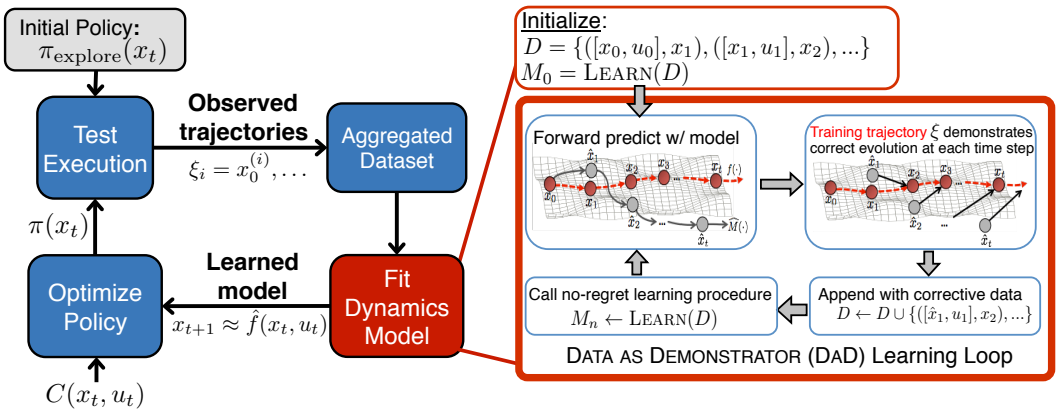


Figure 8.1. (Left) DAgger System Identification for model learning and control. (Right) DAD improves the multi-step prediction performance of learned dynamics models.

successfully proposed in literature. Unfortunately, despite the flexibility of these solutions, the accuracy of data-driven approaches typically depends on the amount of data collected. However, data collection is often expensive or labor intensive in the case of robotics problems, where it is often desirable to observe only few examples on the physical system.

In this chapter, we address the problem of learning improved dynamics models, that can be used for policy learning, with small amount of data. To this end, throughout this chapter we use the Markov Decision Process notation (Definition 2.14), by considering a category of problems in which (1) the transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, or more simply $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, is unknown and (2) a reward $R(s, a)$ or, equivalently, a cost $C(s, a)$ is observed. As introduced in Chapter I, solving an MDP consists of finding a policy π that maximizes the expected cumulative reward, or equivalently, minimizes the expected cumulative cost. In the case of unknown dynamics, approaches based on model-based reinforcement learning (MBRL) typically associate the search for a policy π with the derivation of an estimator $\hat{T}$ of the unknown transition function. This, in fact, is generally known as a system identification problem, that we further discuss in detail to clarify our contribution.

8.1.1. System Identification

System identification has been studied both in the traditional control literature [103] and in the machine learning community [45, 132]. Some of these approaches [132] provide performance guarantees in the case of infinite-data and underlying linear model, while others [45, 69] optimize the the single-step predictive criterion

$$\hat{T} = \arg\min \sum_{t=1}^{T-1} \|s_t - \hat{T}(s_{t-1}, a_{t-1})\|_2^2 \tag{8.1}$$

from a dataset of trajectories $\{(s_0, a_0) \dots, (s_{T-1}, a_{T-1})\}$ of state-action pairs collected from the system. Unfortunately, using dynamics models learned through Eq. 8.1 typically leads to exponential cascading errors [144]) that make the learned

$\hat{T}$ only partially usable. In particular, simply collecting system trajectories, learning the dynamics, and optimizing the control policy results in inaccurate or unstable dynamics models and poorly performing control policies.

To tackle this problem, an iterative process has been formalized [1, 30], that alternates between fitting the dynamics model and collecting new data under the distribution induced by the policy. Intuitively, on the one hand this mechanism enables the model to improve its estimate of the dynamics over important regions of the state-space (i.e., those visited by the policy); on the other hand, the control policy, that is derived from the dynamics, improves or exploits inaccuracies in the dynamics model. Thus, at each iteration of the algorithm, either a good policy is found, or new data is collected for improvement in the subsequent iteration. A slightly different version of this procedure is represented by the DAgger system identification framework [121] (Figure 8.1). DAgger typically takes as input an empty dataset and an exploration policy $\pi_{\text{explore}}(s_t)$, that can be initialized to be random or by means of expert demonstrations. Then, the algorithm (1) executes the policy to collect a set of state-action pair trajectories $\{\xi_i\}$, where $\xi_i = \{(s_t, a_t)\}_i$; (2) aggregates $\{\xi_i\}$ into the training dataset; (3) learns from the dataset a dynamics model $\hat{T}(s_t, a_t) \rightarrow s_{t+1}$; (4) optimizes a new control π that minimizes $c(s_t, a_t)$ over the time horizon T of the control problem; (5) tracks the best policy from all those generated. This iterative procedure refines the dynamics model by aggregating data generated from states induced by running the system with $\pi_1, \ldots, \pi_N$.

Still, despite the use of iterative procedures, MBRL methods generally suffer from compounding errors during policy optimization, that are due to sequential predictions performed with the learned model. In fact, by performing sequential predictions, the model is recursively applied and its previous output is fed as its new input (Eq. 8.2), thus resulting in a significant deviation from the true system.

$$\hat{s}_{t+1} = \hat{T}(\hat{s}_t, a_t). \tag{8.2}$$

To address this problem, in the next section we describe a model-based reinforcement learning framework that reuses collected data to improve the learned dynamics model.

8.2. Multi-Step Predictive Performance Improvement

In this section we describe Data as Demonstrator for Control, to tackle the dynamics learning problem. For achieving good multi-step predictive performance while using supervised learning methods, Venkatraman et al. [144] recently introduced Data As Demonstrator (DaD). DaD is a meta-algorithm that augments the traditional dynamics learning method with an additional iterative procedure. In particular, to minimize the cascading error, DaD specifically targets the distribution induced from sequential application of the model. To this end, the algorithm performs "rollouts" with the learned model using trajectories from the training data. Then, the algorithm generates synthetic data by creating new input-target pairs that point each prediction to the correct time-indexed state along the training trajectory.[1]

[1] Trajectories can be sub-sampled shorter than the control problem's time horizon.

Input: N: number of iterations N; $\{\xi_k\}$: K trajectories of lengths $\{T_k\}$.
Output: $\hat{T}_n$: dynamics model with the lowest error on validation dataset.

Initialize dataset $\mathcal{D} \leftarrow \{([s_t, a_t], s_{t+1})\}$ from example trajectories ξ_k

`// Iterate the algorithm`
for $n = 1 \dots N$ **do**
  `// Learn a dynamics model from the available dataset`
  $\hat{T}_n \leftarrow learn(\mathcal{D})$
  `// Iterate over the example trajectories`
  for $k = 1 \dots K$ **do**
    `// Extract the initial state from the trajectory`
    $s_0 \leftarrow \xi_k(0)$
    `// Extract the full sequence of actions executed in the trajectory`
    $\{a_t\} \leftarrow \xi_k$
    `// Apply the original sequence of actions to the learned model, to obtain new states`
    $(\widehat{s}_1 \dots \widehat{s}_T) \leftarrow \text{rollout}(\hat{T}_n,\, s_0,\, \{a_t\})$
    `// Use the generated states, the actions, and the true states to create a new dataset`
    $\mathcal{D}_{new} \leftarrow \{([\widehat{s}_1, a_1], s_2) \dots ([\widehat{s}_{T_k-1}, a_{T_k-1}], s_{T_k})\}$ where $s_t \leftarrow \xi_k(t)$
    `// Aggregate the dataset`
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{new}$
  end
end

return $\hat{T}_n$ with lowest error on validation trajectories

Algorithm 9: Data As Demonstrator (DaD) for Control.

While we refer the reader to Venkatraman et al. [144] for theoretical details, DaD (as presented in Venkatraman et al. [144]) only handles uncontrolled dynamics. Here we introduce an extension to this algorithm that enables it to handle controlled systems and to be used in the MBRL setting, as shown on the right side of Figure 8.1.

8.2.1. Data as Demonstrator for Control

Data as Demonstrator for Control is an iterative algorithm that builds on DaD and extends it to controlled dynamics setting. Our method, as detailed in Algorithm 9, relies on data aggregation and, at each iteration, it proceeds as follows:

1. learns a forward dynamics by optimizing a supervised learning loss to predict targets s_{t+1} from "features" $[s_t, a_t]$;

2. executes a rollout of the model to obtain states $\hat{s}_i$. In particular, it starts from a state s_0 taken from the of the trajectory ξ_k and performs forward simulations by means of recursive updates (Eq. 8.2) with (1) the learned model $\hat{T}_n$ and (2) the true sequence of controls $\{a_t\}$ from ξ_k;

3. augments the dataset $\mathcal{D}$ by creating input-target pairs $([\hat{s}_t, a_t], s_{t+1})$. Differently from Venkatraman et al. [144], we do not separate the state transition dynamics from the controls but we jointly optimize the model.

Intuitively, this procedure provides synthetic recovery examples to the learner, which are then used to compensate for the compounding error resulting from recursive updates.

8.3. Experimental Evaluation

In this section, we evaluate our algorithm ('DAgger +DaD') both on simulated dynamical systems[2] and real robotic platforms. In particular, we consider two simulated scenarios: the classic cartpole swing-up problem and the challenging helicopter hovering problem. Additionally, we show the applicability of our approach on real systems such as the Videre Erratic mobile base and the Baxter robot. In each described experiment, we learn dynamical models of the form:

$$\Delta_t \leftarrow T(s_t, a_t), \quad \text{where } \Delta_t = s_{t+1} - s_t. \tag{8.3}$$

This parametrization is similar to [30], where the previous state is used as the mean prior for predicting the next state. Due to the difficulty of optimizing the expected cumulative cost under arbitrary dynamics and cost models, for simplicity, we focus on minimizing a sum-of-quadratics cost-to-go function:

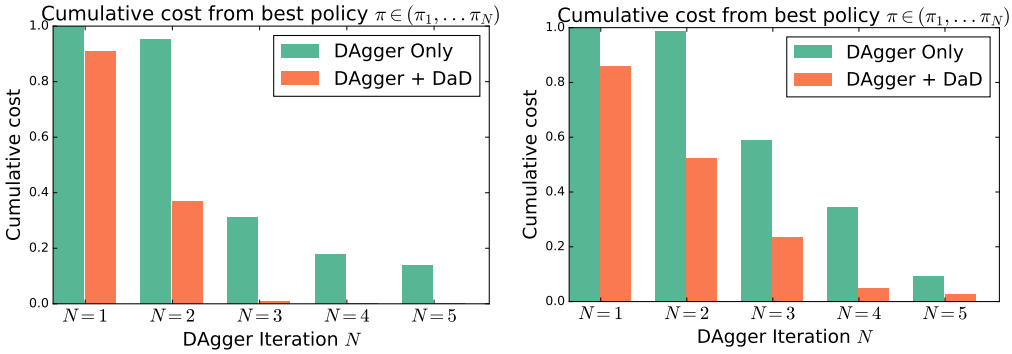
$$\sum_t c(s_t, a_t) = \sum_t s_t^T Q_t s_t + a_t^T R_t a_t. \tag{8.4}$$

By using this form of cost function, along with a linearization of the learned dynamics model, we can formulate the policy synthesis problem as that of a Linear Quadratic Regulator (as introduced in Section 2.4.3), which allows the policy to be computed in closed-form. In each experiment, we compare 'DaD +DAgger' to 'DAgger Only'. For the Cartpole, Erratic, and Baxter experiments, the dataset was initialized with a random exploratory policy, while the helicopter problem received both a random and an expert policy (generated form LQR on the true dynamics). In addition, the simulated cartpole and helicopter experiments collected additional exploratory rollouts on every iteration of DAgger with the random and expert policies respectively. For the Baxter robot, we achieved exploration through an ϵ-random controller that added random perturbation to the commanded control with ϵ probability. For each method, we report the average cumulative cost averaged over ran trials. In particular, we ran three trials on the Erratic and five trials for other benchmarks.

8.3.1. Simulation Experiments

Cartpole swing-up : The cartpole swing-up is a classic controls and MBRL benchmark, where the goal is to swing-up a pendulum by only applying a linear force on the translatable base. We learn a linear dynamics model in the form of Eq. 8.3 using Ridge Regression (regularized linear regression). We then use an iterative Linear Quadratic Regulator [81] (iLQR) controller about a nominal swing-up trajectory in state-space with an initial control trajectory of zeros. The iLQR optimization procedure finds a sequence of states and controls feasible under the

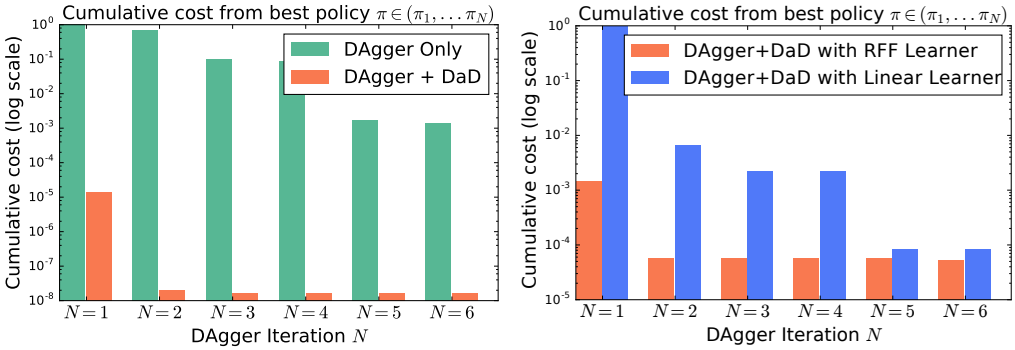
[2]Simulators, except the helicopter, available at `https://github.com/webrot9/control_simulators` with C++ and Python APIs.



(a) Cartpole swing-up.

(b) Cartpole swing-up with control noise.

Figure 8.2. Results on simulated cartpole controlled for swing-up behavior through iLQR about nominal trajectory.



(a) Helicopter hover.

(b) Helicopter hover with different learning algorithms.

Figure 8.3. Results on simulated helicopter controlled for hover behavior (the cost is expressed in log-scale).

learned dynamics model to minimize the cost. The simulated system has system-transition noise. In addition, we compare our algorithm's performance both with, and without control noise – to simulate the effects of noisy actuation on a real-robot. We show results in Figure 8.2 of the evaluated trajectory costs, accumulated over the problem's time horizon.

Helicopter simulator : Helicopter hovering is a difficult problem due to the instability of the dynamical system, especially under noise. We utilize the helicopter simulator from [1] with additive white noise and follow a problem setup similar to [121]. In addition, we make the problem more difficult by initializing the helicopter at states up to 10 meters away from the nominal hover configuration. As the dynamics are highly non-linear, we show the advantage of using Random Fourier Features (RFF) regression [112] to learn a dynamics model in a 21-dimensional state space. We find a steady-state linear quadratic regulator (LQR) policy to map the helicopter's

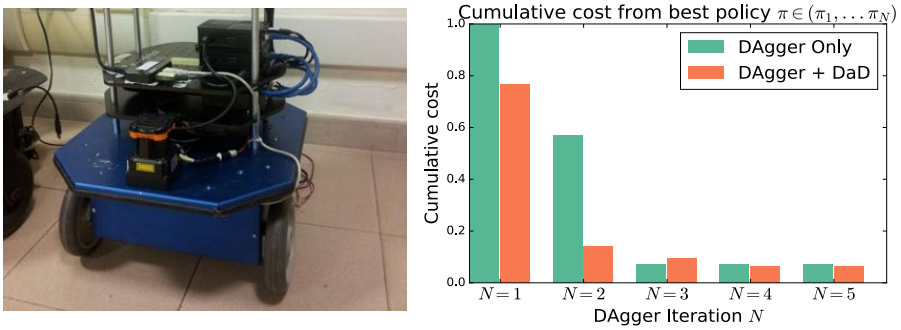


Figure 8.4. Results for controlling a Videre Erratic differential-drive mobile robot in open-loop.

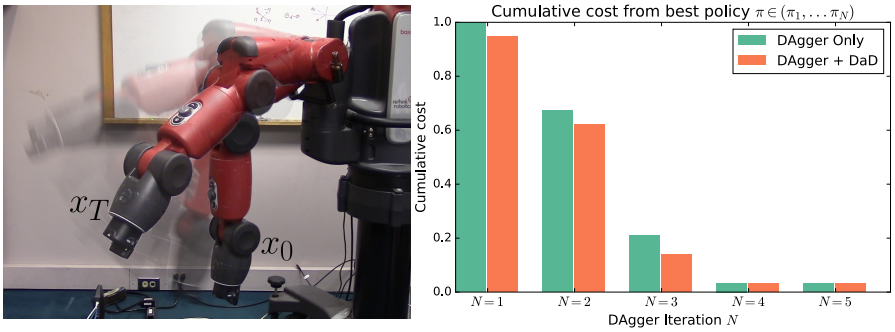


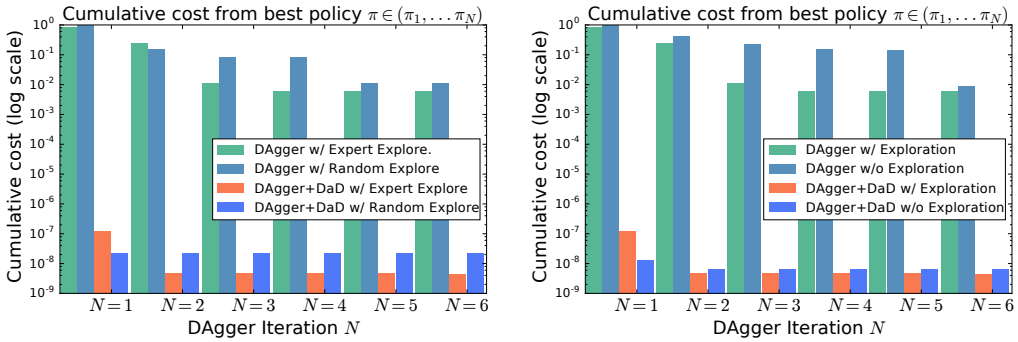
Figure 8.5. Results on controlling a Baxter robot with a steady-state control policy. We learn a dynamics model and compute a policy to move the robot manipulator from state s_0 to s_T.

state to the 4-D control input. The results in Figure. 8.3 show that DAD dramatically improves performance over only DAgger.

8.3.2. Real-Robot Experiments

Videre Erratic: In this experiment, we control the velocity of a Videre Erratic mobile base. The goal is to drive the robot to a given position specified in the robot's reference frame. The 3-D state vector includes the robot position and orientation while the 2-D control vector is the robot velocity. The dynamics model is learned using Ridge Regression. Unlike other experiments, we use a trajectory-control policy that finds a sequence of controls $a_1, \ldots, a_T$ to be applied in open-loop, at run-time, on the robot. We compute the control sequence by simulating the learned dynamics model $\hat{T}$ with a simple proportional controller. Results are shown in Figure 8.4.

Baxter robot: We use the 'DAgger +DAD' approach to control a 7-degree-of-freedom manipulator to a target joint configuration. We command the robot arm in torque control mode with suppression of the inbuilt gravity compensation. The 14-dimensional state vector consists of the joint angles and their velocities. We learn the dynamics model using Ridge Regression and compute a steady-state LQR control policy, obtaining the results in Fig. 8.5.



(a) Random vs. expert exploration policies. **(b)** Exploration vs no exploration policy.

Figure 8.6. Comparison of exploration policies. Cost values are not normalized across plots.

8.3.3. Reflections on the Experiments

In our simulation experiments we compared the performance obtained by applying 'DAgger +DaD' on a cartpole with and without control noise. Results show that the improvement of our method over 'DAgger Only' decreases in presence of actuation noise. This can be explained by the fact that, over the same generated nominal controls, the state trajectories obtained during each rollout are slightly different and represent a limitation on the efficacy of the learner over the same number of iterations – i.e. there is a higher baseline error in the dynamics model.

In the case of the helicopter, we additionally compared the results obtained by using two different learning algorithms and by applying different exploration policies. For the former, we compared the non-linear RFF [112] regression against linear regression. As shown in Figure 8.3b, the non-linear learner obtains a better result, thus capturing the heavy non-linearity of the helicopter dynamics. The DAgger method [121] requires drawing state-transition samples at every iteration from some exploration distribution. In Figure 8.6a, we compare using an expert exploration policy (LQR controller using the true dynamics) versus a random-control exploration policy. With 'DAgger +DaD', the learned dynamics and policy yield a stable behavior for both types of exploration, with some improvement using the expert policy. The 'DAgger Only' baseline often is unable to learn a stable policy using the random exploration policy. Interestingly, 'DAgger +DaD' without the exploration policy does not lead to a significant performance difference (Figure 8.6b) compared to the baselines. This comparison, in particular, shows the difference between Abbeel and Ng [1] (no exploration) and Ross and Bagnell [121] (constant fraction exploration).

The real-robot evaluations show the applicability of our method on real systems and complex platforms. In particular, the Erratic experiments show that by using DaD, we are indeed able to get a better dynamics model for forward-prediction. This model can be used for trajectory generation and optimization as described in Section 8.3.2, where the sequence of obtained controls has been directly applied to the Erratic in an open-loop as a control trajectory. While the application of 'DAgger +DaD' on the Baxter robot results in a limited performance improvement, this

confirms our hypothesis that, in robotic platforms characterized by high actuation noise (e.g. Baxter's chain of noisy actuators), only smaller improvements over 'DAgger Only' can be achieved. In fact, this is consistent with the simulated noisy-actuation result in Figure 8.2b.

8.4. Contributions

In this chapter, we discussed the issues related to model based reinforcement learning in presence of complex or unknown system dynamics (Section 8.1). Of importance for the goals of this thesis, we introduced an algorithm that extends previous work [144] and enables more accurate forward-simulations of dynamics models (Section 8.2). These models can be then used in the context of our algorithms for situated behavior generation, such as MCSDA and, in particular, π-STAM. In fact, results illustrated in Section 8.3 show the efficacy of our algorithm in developing LQR, iLQR, and open-loop trajectory-based control strategies both on simulated benchmarks as well as physical robot platforms.

Part IV

Conclusions

Chapter 9

Conclusions and Discussion

> *The end of a melody is not its goal: but nonetheless, had the melody not reached its end it would not have reached its goal either. A parable.*
>
> — Friedrich Nietzsche

The main premise of our work has been that spatial semantics is necessary to obtain accurate and intuitive world models, that enable both deliberation and situated activity. In this chapter, we briefly discuss and summarize our main theoretical and practical findings in support of that claim. Finally, we briefly discuss some open research problems and possible directions for extending our work.

9.1. Summary of Contributions

In this section, we summarize the contributions and novelties introduced in this thesis in relation to the specific purposes for which they have been adopted.

9.1.1. Models for Spatial Semantics

The role of spatial semantics has been investigated by first introducing two models for describing environmental knowledge. In particular, we formalized a novel representation that explicitly models the environment through the notion of semantic maps. These maps intuitively formalize high-level knowledge and symbolic information. Then, we formalized a new model of spatio-temporal affordances. This model represents action semantics in relation to the environment and implicitly encodes the effects of the actions. We leverage these models throughout this thesis to generate our semantic-driven robot behaviors.

9.1.2. Action Semantics and Policy Generation

In the statement of our thesis, we argued that spatial semantics enable both deliberation and situated activity. To support this argument, we used our two models of spatial semantics to generate both high-level behaviors, by using semantic maps,

and situated policies, by means of the spatio-temporal affordance model. In particular, in the former case we directly leverage traditional reasoning and inference mechanisms. In latter case, instead, we adopt novel tools for policy learning and system identification through the MDP formalism.

9.1.3. Applications and Empirical Findings

By combining these models, together with our algorithms, we have been able to effectively generate robot behaviors that leverage semantic information. In particular, we obtained high-level and intuitive policies that enable natural communication with human agents. These policies have been then used on multiple instances of "service robots". Moreover, we generated situated agent's behaviors that explicitly account for action semantics. We applied this principle in the context of decision making with humanoids robots.

To summarize, we have shown novel results on the formalization of models that account for semantic information. Additionally, we have shown that these models can be effectively used to affect the generation and learning of robot behaviors at different scales and multiple contexts.

9.2. Open Problems and Future Directions

In this section, we briefly discuss some open research problems and possible directions for extending our work. These extensions, in our opinion, go primarily in two directions. The former is related to understanding what is a "good" semantic-driven system, and how that can be measured. The second accounts for the difficulties of effectively explicating semantic knowledge during learning processes that generate agent behaviors.

9.2.1. Benchmarking of Semantic-Driven Systems

Throughout this thesis we have repeatedly argued that comparing and evaluating semantic maps is a difficult task. On the one side, there is a structural limit, which is determined by the multiplicity of representations and algorithms that are proposed in literature. On the other side, there is a substantial limit, that is represented by the absence of benchmarking datasets and common testing environments. In this thesis we attempted to address these issues, by proposing a basic formal model for representing semantic maps and for acquiring knowledge in a principled way. Additionally, we proposed the creation of a semantic mapping dataset and illustrated how that can be generated in accordance to the proposed formal model. On this side, however, multiple questions remain still open. For example:

- how can we fairly compare autonomous semantic mapping systems, against those that leverage human intervention?
- should we evaluate semantic maps according to task oriented metrics, or according to metrics that evaluate spatial properties (e.g, the distance with respect to a ground-truth)?

- can semantic maps, as explicit representations of knowledge, be compared against implicit representations?

Unfortunately, similar underlying issues extend to alternative approaches that attempt to model semantic information. This is particularly true, for example, in the case of affordance models. Affordances, in fact, are based on the theories of ecological psychology, and represent the semantics of actions as explored subjectively by each agent during its "life" within the environment. In this sense, there is not an absolute ground-truth, and quantitative evaluations are difficult and almost restrictive. Establishing some evaluation criteria that account for these issues, then, is an open problem of potentially increasing interest for the robotics community.

9.2.2. Semantic-Driven Learning in Robotics

In our work, we use spatio-temporal affordance maps to capture action semantics, as explored by the agent, and simultaneously restrict the policy search space by means of the learned model. While this is an effective solution for relatively simple problems, this type of approach does not scale to complex and hierarchical tasks. Multiple questions, in fact, are left open under this setting:

- how do affordances interact among themselves, when organized on multiple levels?
- how can affordances be used for the realization of a continual learning algorithm?
- is there space for important improvements, by using semantic-driven learning at different scales?

In this thesis, we only investigated how affordances can be used for situated actions, while semantic maps can be used for deliberation. Is it possible to use a unique representation, to account for the semantics of the environment at both scales? If yes, how can affordance models be used for learning more abstract information? Multiple questions open in this direction, leaving the space for future research and interesting considerations.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 1–8, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102352. (Cited on pages 109, 112, and 114.)

[2] R.C. Arkin. *Behavior-based Robotics*. Bradford book. MIT Press, 1998. ISBN 9780262011655. (Cited on page 5.)

[3] Ronald C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6(1):105 – 122, 1990. ISSN 0921-8890. doi: 10.1016/S0921-8890(05)80031-4. (Cited on pages 5 and 6.)

[4] T. Asfour, F. Gyarfas, P. Azad, and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, pages 40–47, Dec 2006. doi: 10.1109/ICHR.2006.321361. (Cited on page 37.)

[5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, May 2002. ISSN 0885-6125. doi: 10.1023/A:1013689704352. (Cited on page 26.)

[6] J. A. Bagnell and J. G. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *2001 IEEE International Conference on Robotics and Automation*, volume 2, pages 1615–1620 vol.2, 2001. doi: 10.1109/ROBOT.2001.932842. (Cited on page 107.)

[7] B. Bakker, V. Zhumatiy, G. Gruener, and J. Schmidhuber. Quasi-online reinforcement learning for robots. In *2006 IEEE International Conference on Robotics and Automation*, pages 2997–3002, May 2006. doi: 10.1109/ROBOT.2006.1642157. (Cited on pages 37 and 107.)

[8] Philippe Balbiani, Jean-François Condotta, and Luis Fariñas del Cerro. A new tractable subclass of the rectangle algebra. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 442–447. Morgan Kaufmann, 1999. ISBN 1-55860-613-0. (Cited on page 33.)

[9] E Bastianelli, DD Bloisi, R Capobianco, F Cossu, G Gemignani, L Iocchi, and D Nardi. On-line semantic mapping. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–6. IEEE, 2013. doi: 10.1109/ICAR.2013.6766501. (Cited on pages 32, 43, and 87.)

[10] Emanuele Bastianelli, Danilo Croce, Andrea Vanzo, Roberto Basili, and Daniele Nardi. A discriminative approach to grounded spoken language understanding in interactive robotics. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, New York, USA, July, 9–15 2016. (Cited on page 3.)

[11] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992. ISSN 0162-8828. doi: 10.1109/34.121791. (Cited on page 78.)

[12] N. Blodow, L. C. Goron, Z. C. Marton, D. Pangercic, T. Rühr, M. Tenorth, and M. Beetz. Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4263–4270, Sept 2011. doi: 10.1109/IROS.2011.6094665. (Cited on page 30.)

[13] D. Boud, R. Keogh, and D. Walker. *Reflection: Turning Experience Into Learning*. Kogan Page, 1985. ISBN 9780850388640. (Cited on pages xiii, 6, and 7.)

[14] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, Mar 1986. ISSN 0882-4967. doi: 10.1109/JRA.1986.1087032. (Cited on page 5.)

[15] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012. ISSN 1943-068X. doi: 10.1109/TCIAIG.2012.2186810. (Cited on pages xiii, 25, 26, 92, and 93.)

[16] E. Brunskill, T. Kollar, and N. Roy. Topological mapping using spectral clustering and classification. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3491–3496, Oct 2007. doi: 10.1109/IROS.2007.4399611. (Cited on page 30.)

[17] Dídac Busquets, Carles Sierra, and Ramon López de Màntaras. A multiagent approach to qualitative landmark-based navigation. *Autonomous Robots*, 15(2):129–154, 2003. ISSN 1573-7527. doi: 10.1023/A:1025536924463. (Cited on page 34.)

[18] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics Automation Magazine*, 17 (2):44–54, June 2010. ISSN 1070-9932. doi: 10.1109/MRA.2010.936947. (Cited on page 37.)

[19] M. Cao, A. Stewart, and N. E. Leonard. Integrating human and robot decision-making dynamics with feedback: Models and convergence analysis. In *47th IEEE Conference on Decision and Control (CDC)*, pages 1127–1132, Dec 2008. doi: 10.1109/CDC.2008.4739103. (Cited on page 6.)

[20] Roberto Capobianco, Guglielmo Gemignani, Domenico Bloisi, Daniele Nardi, and Luca Iocchi. Automatic extraction of structural representations of environments. In *Intelligent Autonomous Systems 13*, pages 721–733. Springer International Publishing, 2014. ISBN 978-3-319-08337-7. doi: 10.1007/978-3-319-08338-4_52. (Cited on pages xiv, 78, and 85.)

[21] Roberto Capobianco, Jacopo Serafin, Johann Dichtl, Giorgio Grisetti, Luca Iocchi, and Daniele Nardi. A proposal for semantic map representation and evaluation. In *2015 European Conference on Mobile Robots*, pages 1–6. IEEE, 2015. doi: 10.1109/ECMR.2015.7324198. (Cited on pages 30, 43, and 59.)

[22] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. Learning to search better than your teacher. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning*, pages 2058–2066. JMLR Workshop and Conference Proceedings, 2015. (Cited on pages 38 and 90.)

[23] Anthony Chemero. An outline of a theory of affordances. *Ecological Psychology*, 15(2): 181–195, 2003. doi: 10.1207/S15326969ECO1502_5. (Cited on pages 20 and 61.)

[24] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 233:1–233:8, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5. doi: 10.1145/1329125.1329407. (Cited on page 37.)

[25] Maurilio Di Cicco, Luca Iocchi, and Giorgio Grisetti. Non-parametric calibration for depth sensors. *Robotics and Autonomous Systems*, 74, Part B:309 – 317, 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2015.08.004. Intelligent Autonomous Systems. (Cited on page 54.)

[26] Simon Coffey and Keith Clark. A hybrid, teleo-reactive architecture for robot control. In *Multi-Agent Robotic Systems*, July 2006. (Cited on pages 5 and 6.)

[27] J. H. Connell. Sss: a hybrid architecture applied to robot navigation. In *1992 IEEE International Conference on Robotics and Automation*, pages 2719–2724 vol.3, May 1992. doi: 10.1109/ROBOT.1992.219995. (Cited on pages 5 and 6.)

[28] Ignasi Cos-Aguilera, Lola Cañamero, and Gillian M Hayes. Motivation-driven learning of object affordances: First experiments using a simulated khepera robot. In Frank Detjer, Dietrich Dörner, and Harald Schaub, editors, *Proceedings of the 5th International Conference in Cognitive Modelling*, pages 57–62, Bamberg, Germany, 2003. (Cited on pages 35 and 61.)

[29] Giuseppe De Giacomo and Maurizio Lenzerini. Tbox and abox reasoning in expressive description logics. In *Proceedings of the 1996 International Workshop on Description Logics*, volume WS-96-05 of *AAAI Technical Report*, pages 37–48. AAAI Press, 1996. ISBN 1-57735-014-6. (Cited on page 19.)

[30] M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, pages 465–472, 2011. (Cited on pages 109 and 111.)

[31] Yong Duan, Qiang Liu, and XinHe Xu. Application of reinforcement learning in robot soccer. *Engineering Applications of Artificial Intelligence*, 20(7):936 – 950, 2007. ISSN 0952-1976. doi: 10.1016/j.engappai.2007.01.003. (Cited on pages 37 and 107.)

[32] Andrew P. Duchon, Leslie Pack Kaelbling, and William H. Warren. Ecological robotics. *Adaptive Behavior*, 6(3-4):473–507, 1998. doi: 10.1177/105971239800600306. (Cited on pages 3 and 20.)

[33] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-56876-5. (Cited on page 18.)

[34] S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 358–363, Sept 2006. doi: 10.1109/ROMAN.2006.314460. (Cited on page 38.)

[35] Susan L. Epstein, Anoop Aroor, Matthew Evanusa, Elizabeth Sklar, and Simon Parsons. Navigation with learned spatial affordances. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*. cognitivesciencesociety.org, 2015. ISBN 978-0-9911967-2-2. (Cited on pages 34, 37, and 61.)

[36] Paul M. Fitzpatrick, Giorgio Metta, Lorenzo Natale, Ajit Rao, and Giulio Sandini. Learning about objects through action -initial steps towards artificial cognition. In *2003 IEEE International Conference on Robotics and Automation*, pages 3140–3145, 2003. doi: 10.1109/ROBOT.2003.1242073. (Cited on page 35.)

[37] Andrew U Frank. Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages & Computing*, 3(4):343 – 371, 1992. ISSN 1045-926X. doi: 10.1016/1045-926X(92)90007-9. (Cited on page 33.)

[38] F.S. Freeman. *Theory and Practice of Psychological Testing*. Holt, 1950. ISBN 9788120417076. doi: 10.1002/1097-4679(195010)6:4<420::aid-jclp2270060431>3.0.co;2-y. (Cited on page 1.)

[39] Stephen Friedman, Hanna Pasula, and Dieter Fox. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2109–2114, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. (Cited on page 30.)

[40] A. L. Friesen and R. P. N. Rao. Imitation learning with hierarchical actions. In *IEEE 9th International Conference on Development and Learning*, pages 263–268, Aug 2010. doi: 10.1109/DEVLRN.2010.5578832. (Cited on page 38.)

[41] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. A. Fernandez-Madrigal, and J. Gonzalez. Multi-hierarchical semantic maps for mobile robotics. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2278–2283, Aug 2005. doi: 10.1109/IROS.2005.1545511. (Cited on pages 30 and 31.)

[42] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 809–815. AAAI Press, 1992. ISBN 0-262-51063-4. (Cited on page 5.)

[43] Guglielmo Gemignani, Roberto Capobianco, Emanuele Bastianelli, Domenico Bloisi, Luca Iocchi, and Daniele Nardi. Living with robots: Interactive environmental knowledge acquisition. *Robotics and Autonomous Systems*, 78:1–16, 2016. doi: 10.1016/j.robot.2015.11.001. (Cited on pages 76, 78, 86, and 87.)

[44] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987. ISBN 0-934613-31-1. (Cited on page 7.)

[45] Zoubin Ghahramani and Sam T. Roweis. Learning nonlinear dynamical systems using an em algorithm. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 431–437, Cambridge, MA, USA, 1999. MIT Press. ISBN 0-262-11245-0. (Cited on page 108.)

[46] James Jerome Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979. ISBN 0-395-27049-9. Includes indexes. (Cited on pages v, 20, 34, and 61.)

[47] Nils Goerke and Sven Braun. Building semantic annotated maps by mobile robots. In *Proceedings of the Conference Towards Autonomous Robotic Systems*, pages 149–156, 2009. (Cited on pages 17 and 30.)

[48] Linda S Gottfredson. Mainstream science on intelligence: An editorial with 52 signatories, history, and bibliography. *Intelligence*, 24(1):13–23, 1997. doi: 10.1016/S0160-2896(97)90011-8. (Cited on page 1.)

[49] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, winter 2010. ISSN 1939-1390. doi: 10.1109/MITS.2010.939925. (Cited on page 55.)

[50] F. Guenter and A. G. Billard. Using reinforcement learning to adapt an imitation task. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1022–1027, Oct 2007. doi: 10.1109/IROS.2007.4399449. (Cited on page 37.)

[51] M Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Building semantic object maps from sparse and noisy 3D data. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2228–2233. IEEE, 2013. doi: 10.1109/IROS.2013.6696668. (Cited on pages 30, 31, and 43.)

[52] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *2014 IEEE International Conference on Robotics and Automation*, pages 1524–1531, 2014. doi: 10.1109/ICRA.2014.6907054. (Cited on page 43.)

[53] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, 1990. doi: 10.1016/0167-2789(90)90087-6. (Cited on page 19.)

[54] S. Hemachandra, T. Kollar, N. Roy, and S. Teller. Following and interpreting narrated guided tours. In *2011 IEEE International Conference on Robotics and Automation*, pages 2574–2579, May 2011. doi: 10.1109/ICRA.2011.5980209. (Cited on pages 30 and 31.)

[55] T. Hester, M. Quinlan, and P. Stone. RTMBA: A real-time model-based reinforcement learning architecture for robot control. In *2012 IEEE International Conference on Robotics and Automation*, pages 85–90, May 2012. doi: 10.1109/ICRA.2012.6225072. (Cited on pages 37 and 107.)

[56] Thomas E Horton, Arpan Chakraborty, and Robert St Amant. Affordances for robots: a brief survey. *Avant*, 3(2):70–84, 2012. (Cited on page 20.)

[57] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003. URL `http://radish.sourceforge.net/`. (Cited on pages xiii and 18.)

[58] Jorge Ibáñez Gijón, Alex Díaz, Lorena Lobo, and David Jacobs. On the ecological approach to information and control for roboticists. *International Journal of Advanced Robotic Systems*, 10(6), 2013. doi: 10.5772/55671. (Cited on page 3.)

[59] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Trajectory formation for imitation with nonlinear dynamical systems. In *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 752–757 vol.2, 2001. doi: 10.1109/IROS.2001.976259. (Cited on page 38.)

[60] Luca Iocchi, Daniele Nardi, and Massimiliano Salerno. *Reactivity and Deliberation: A Survey on Multi-Robot Systems*, pages 9–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-44568-5. doi: 10.1007/3-540-44568-4_2. (Cited on page 5.)

[61] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047270. (Cited on page 78.)

[62] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–91, 1979. ISSN 00129682, 14680262. doi: 10.2307/1914185. (Cited on page 6.)

[63] Marc Kammer, Thomas Schack, Marko Tscherepanow, and Yukie Nagai. From affordances to situated affordances in robotics - why context is important. *Frontiers in Computational Neuroscience*, 5:30, 2011. ISSN 1662-5188. doi: 10.3389/conf.fncom.2011.52.00030. (Cited on page 36.)

[64] Mubbasir Kapadia, Shawn Singh, William Hewlett, and Petros Faloutsos. Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 215–223, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-429-4. doi: 10.1145/1507149.1507185. (Cited on page 36.)

[65] Dov Katz, Arun Venkatraman, Moslem Kazemi, J. Andrew Bagnell, and Anthony Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots*, 37(4):369–382, 2014. ISSN 1573-7527. doi: 10.1007/s10514-014-9407-y. (Cited on page 39.)

[66] Piyush Khandelwal and Peter Stone. Multi-robot human guidance using topological graphs. In *AAAI Spring 2014 Symposium on Qualitative Representations for Robots (AAAI-SSS)*, March 2014. (Cited on pages xiii and 19.)

[67] D. I. Kim and G. S. Sukhatme. Interactive affordance map building for a robotic task. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4581–4586, Sept 2015. doi: 10.1109/IROS.2015.7354029. (Cited on pages xiii, 34, 35, 36, and 61.)

[68] Matthew Klenk, Nick Hawes, and Kate Lockwood. Representing and reasoning about spatial regions defined by context. In *AAAI Fall Symposium 2011 on Advances in Cognitive Systems*, 2011. (Cited on page 34.)

[69] J. Ko, D. J. Kleint, D. Fox, and D. Haehnelt. Gp-ukf: Unscented kalman filters with gaussian process prediction and observation models. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1901–1907, Oct 2007. doi: 10.1109/IROS.2007.4399284. (Cited on pages 107 and 108.)

[70] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1):171–203, 2011. ISSN 1573-0565. doi: 10.1007/s10994-010-5223-6. (Cited on page 38.)

[71] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal Of Robotics Research*, 32(11):1238–1274, 2013. doi: 10.1177/0278364913495721. (Cited on page 37.)

[72] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research*, 31(3):360–375, March 2012. ISSN 0278-3649. doi: 10.1177/0278364911428653. (Cited on pages 37 and 107.)

[73] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, 32(8):951–970, July 2013. ISSN 0278-3649. doi: 10.1177/0278364913478446. (Cited on pages 34 and 35.)

[74] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, Oct 2010. doi: 10.1109/IROS.2010.5649089. (Cited on page 38.)

[75] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86 – 103, 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2014.12.006. (Cited on pages 17 and 30.)

[76] Geert-Jan M. Kruijff, Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. Clarification dialogues in human-augmented mapping. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, HRI '06, pages 282–289, New York, NY, USA, 2006. ACM. ISBN 1-59593-294-1. doi: 10.1145/1121241.1121290. (Cited on pages 30 and 31.)

[77] Benjamin Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2): 191–233, May 2000. ISSN 0004-3702. doi: 10.1016/S0004-3702(00)00017-5. (Cited on page 29.)

[78] Lars Kunze, Chris Burbridge, and Nick Hawes. Bootstrapping probabilistic models of qualitative spatial relations for active visual object search. In *AAAI Spring Symposium 2014 on Qualitative Representations for Robots*, Stanford University in Palo Alto, California, US, March, 24–26 2014. (Cited on page 34.)

[79] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. ISSN 1532-4435. (Cited on page 2.)

[80] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. In *International Symposium on Experimental Robotics*, 2016. (Cited on page 2.)

[81] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, pages 222–229, 2004. ISBN 972-8865-12-0. doi: 10.5220/0001143902220229. (Cited on page 111.)

[82] M. Lopes, F. S. Melo, L. Montesano, F. Guenter, and A. G. Billard. Affordance-based imitation learning in robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1015–1021, Oct 2007. doi: 10.1109/IROS.2007.4399517. (Cited on page 39.)

[83] Matthias Luber, Gian Diego Tipaldi, and Kai O Arras. Place-dependent people tracking. *The International Journal of Robotics Research*, 30(3):280–293, March 2011. ISSN 0278-3649. doi: 10.1177/0278364910393538. (Cited on pages 34 and 36.)

[84] Pattie Maes and Daniele Nardi. *Meta-Level Architectures and Reflection*. Elsevier Science Inc., New York, NY, USA, 1988. ISBN 0444703438. (Cited on page 7.)

[85] Maja J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997. ISSN 1573-7527. doi: 10.1023/A:1008819414322. (Cited on pages 37 and 107.)

[86] G. Metta and P. Fitzpatrick. Early integration of vision and manipulation. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pages 2703 vol.4–, July 2003. doi: 10.1109/IJCNN.2003.1223994. (Cited on page 35.)

[87] Giorgio Metta, Giulio Sandini, Lorenzo Natale, and Francesco Panerai. Development and robotics. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 33–42, 2001. (Cited on page 35.)

[88] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. Technical report, DeepMind, 2016. (Cited on page 2.)

[89] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015. doi: 10.1038/nature14236. (Cited on pages 2, 37, and 38.)

[90] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Affordances, development and imitation. In *IEEE 6th International Conference on Development and Learning*, pages 270–275, July 2007. doi: 10.1109/DEVLRN.2007.4354054. (Cited on page 35.)

[91] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, Feb 2008. ISSN 1552-3098. doi: 10.1109/TRO.2007.914848. (Cited on pages 35 and 61.)

[92] Oscar Martinez Mozos, Hitoshi Mizutani, Ryo Kurazume, and Tsutomu Hasegawa. Categorization of indoor places using the Kinect sensor. *Sensors*, 12(12):6695–6711, May 2012. ISSN 1424-8220. doi: 10.3390/s120506695. (Cited on pages 17, 30, and 31.)

[93] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989. ISSN 0018-9219. doi: 10.1109/5.24143. (Cited on page 21.)

[94] R. R. Murphy. Case studies of applying gibson's ecological approach to mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 29(1):105–111, Jan 1999. ISSN 1083-4427. doi: 10.1109/3468.736365. (Cited on pages 35 and 61.)

[95] Mark A. Musen. The Protégé project: A look back and a look forward. *AI Matters*, 1 (4):4–12, June 2015. ISSN 2372-3483. doi: 10.1145/2757001.2757003. (Cited on pages xiii, 44, and 55.)

[96] Daniele Nardi, Ronald J Brachman, et al. An introduction to description logics. In *The Description Logic Handbook*, pages 1–40. Cambridge Univ. Press, Cambridge, 2003. ISBN 9780521150118. (Cited on page 19.)

[97] Gerhard Neumann, Christian Daniel, Alexandros Paraschos, Andras Kupcsik, and Jan Peters. Learning modular policies for robotics. *Frontiers in Computational Neuroscience*, 8:62, 2014. ISSN 1662-5188. doi: 10.3389/fncom.2014.00062. (Cited on page 38.)

[98] Monica N. Nicolescu and Maja J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 241–248, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8. doi: 10.1145/860575.860614. (Cited on page 37.)

[99] C. Nieto-Granda, J. G. Rogers, A. J. B. Trevor, and H. I. Christensen. Semantic map partitioning in indoor environments using regional analysis. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1451–1456, Oct 2010. doi: 10.1109/IROS.2010.5650575. (Cited on pages 30 and 31.)

[100] Nils J. Nilsson. Shakey the robot. Technical Report 323, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, Apr 1984. (Cited on page 5.)

[101] D.A. Norman. *The Design of Everyday Things*. Number No. 842 in The Design of Everyday Things. Doubleday, 1988. ISBN 9780385267748. (Cited on pages 21 and 34.)

[102] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008. doi: 10.1016/j.robot.2008.08.001. (Cited on pages v, 3, 9, 17, 30, and 43.)

[103] Peter Van Overschee and Bart De Moor. N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75 – 93, 1994. ISSN 0005-1098. doi: 10.1016/0005-1098(94)90230-5. (Cited on page 108.)

[104] A.K. Pandey and R. Alami. Taskability graph: Towards analyzing effort based agent-agent affordances. In *The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 791–796, Sept 2012. doi: 10.1109/ROMAN.2012.6343848. (Cited on pages 34 and 36.)

[105] D. Pangercic, B. Pitzer, M. Tenorth, and M. Beetz. Semantic object maps for robotic housework - representation, acquisition and use. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4644–4651, Oct 2012. doi: 10.1109/IROS.2012.6385603. (Cited on pages xiii, 31, and 33.)

[106] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zollner. Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):322–332, April 2007. ISSN 1083-4419. doi: 10.1109/TSMCB.2006.886951. (Cited on page 38.)

[107] Michael Pardowitz, Raoul Zöllner, and Rudiger Dillmann. *Incremental Learning of Task Sequences with Information-Theoretic Metrics*, pages 51–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32689-2. doi: 10.1007/11681120_5. (Cited on page 38.)

[108] J. Peltason, F. H. K. Siepmann, T. P. Spexard, B. Wrede, M. Hanheide, and E. A. Topp. Mixed-initiative in human augmented mapping. In *2009 IEEE International Conference on Robotics and Automation*, pages 2146–2153, May 2009. doi: 10.1109/ROBOT.2009.5152683. (Cited on pages 30 and 31.)

[109] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *2012 IEEE International Conference on Robotics and Automation*, pages 3515–3522. IEEE, 2012. doi: 10.1109/ICRA.2012.6224637. (Cited on pages xiii, 2, 30, 31, 32, and 43.)

[110] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471619779. doi: 10.1002/9780470316887. (Cited on page 1.)

[111] Faisal Qureshi, Demetri Terzopoulos, and Ross Gillett. *The Cognitive Controller: A Hybrid, Deliberative/Reactive Control Architecture for Autonomous Robots*, pages 1102–1111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24677-0. doi: 10.1007/978-3-540-24677-0_113. (Cited on pages 5 and 6.)

[112] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008. (Cited on pages 112 and 114.)

[113] Gabriele Randelli, Taigo Maria Bonanni, Luca Iocchi, and Daniele Nardi. Knowledge acquisition through human–robot multimodal interaction. *Intelligent Service Robotics*, 6(1):19–31, 2013. ISSN 1861-2784. doi: 10.1007/s11370-012-0123-1. (Cited on pages 30 and 31.)

[114] Nathan Ratliff. *Learning to Search: Structured Prediction Techniques for Imitation Learning*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2009. (Cited on page 1.)

[115] L. Riazuelo, M. Tenorth, D. Di Marco, M. Salas, D. Gálvez-López, L. Mösenlechner, L. Kunze, M. Beetz, J. D. Tardós, L. Montano, and J. M. M. Montiel. Roboearth semantic mapping: A cloud enabled knowledge-based approach. *IEEE Transactions on Automation Science and Engineering*, 12(2):432–443, 2015. doi: 10.1109/TASE.2014.2377791. (Cited on pages 2 and 43.)

[116] Francesco Riccio, Roberto Capobianco, Marc Hanheide, and Daniele Nardi. *STAM: A Framework for Spatio-Temporal Affordance Maps*, pages 271–280. Springer International Publishing, Cham, 2016. ISBN 978-3-319-47605-6. doi: 10.1007/978-3-319-47605-6_22. (Cited on page 70.)

[117] Francesco Riccio, Roberto Capobianco, and Daniele Nardi. Learning human-robot handovers through π-STAM: Policy improvement with spatio-temporal affordance maps. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2016. (Cited on pages 34 and 105.)

[118] B. Ridge, Danijel Skočaj, and Aleš Leonardis. A system for learning basic object affordances using a self-organizing map. In *International Conference on Cognitive Systems (CogSys)*, 2008. (Cited on page 35.)

[119] Mark B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, Austin, Texas 78712, August 1994. (Cited on page 38.)

[120] Stephane Ross. *Interactive Learning for Sequential Decisions and Predictions*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, June 2013. (Cited on pages 1, 37, and 38.)

[121] Stephane Ross and Drew Bagnell. Agnostic system identification for model-based reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1703–1710, 2012. (Cited on pages 109, 112, and 114.)

[122] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014. (Cited on pages 38 and 90.)

[123] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011. (Cited on pages 37, 38, 90, and 91.)

[124] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010. ISBN 978-0-13-207148-2. (Cited on page 19.)

[125] Erol Şahin, Maya Çakmak, Mehmet R Doğar, Emre Uğur, and Göktürk Üçoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472, 2007. doi: 10.1177/1059712307084689. (Cited on pages 34 and 35.)

[126] Paulo E. Santos, Murilo F. Martins, Valquiria Fenelon, Fabio G. Cozman, and Hannah M. Dee. Probabilistic self-localisation on a qualitative map based on occlusions. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(5):781–799, 2016. doi: 10.1080/0952813X.2015.1132265. (Cited on page 34.)

[127] Tom Schaul and Mark B Ring. Better generalization with forecasts. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, Beijing, China, 2013. (Cited on page 66.)

[128] Paul J. H. Schoemaker. The expected utility model: Its variants, purposes, evidence and limitations. *Journal of Economic Literature*, 20(2):529–563, 1982. ISSN 00220515. (Cited on page 6.)

[129] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009. doi: 10.15607/RSS.2009.V.021. (Cited on page 78.)

[130] Jacopo Serafin and Giorgio Grisetti. *Using Augmented Measurements to Improve the Convergence of ICP*, pages 566–577. Springer International Publishing, Cham, 2014. ISBN 978-3-319-11900-7. doi: 10.1007/978-3-319-11900-7_48. (Cited on page 54.)

[131] Jacopo Serafin and Giorgio Grisetti. NICP: Dense normal based point cloud registration. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 742–749. IEEE, 12 2015. ISBN 9781479999941. doi: 10.1109/IROS.2015.7353455. (Cited on pages 54 and 78.)

[132] Sajid M. Siddiqi, Byron Boots, and Geoffrey J. Gordon. A constraint generation approach to learning stable linear dynamical systems. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems*, pages 1329–1336. Curran Associates, Inc., 2007. (Cited on page 108.)

[133] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. doi: 10.1038/nature16961. (Cited on pages 25, 37, and 38.)

[134] Kristoffer Sjöö, Alper Aydemir, and Patric Jensfelt. Topological spatial relations for active visual search. *Robotics and Autonomous Systems*, 60(9):1093–1107, September 2012. ISSN 0921-8890. doi: 10.1016/j.robot.2012.06.001. (Cited on page 34.)

[135] Robert J Sternberg and Scott Barry Kaufman. *The Cambridge Handbook of Intelligence*. Cambridge University Press, 2011. ISBN 9780511977244. doi: 10.1017/CBO9780511977244. (Cited on page 1.)

[136] A. Stoytchev. Behavior-grounded representation of tool affordances. In *2005 IEEE International Conference on Robotics and Automation*, pages 3060–3065, April 2005. doi: 10.1109/ROBOT.2005.1570580. (Cited on page 35.)

[137] L.A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Learning in Doing: Social, Cognitive and Computational Perspectives. Cambridge University Press, 1987. ISBN 9780521337397. (Cited on page 5.)

[138] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981. (Cited on pages xiii, 23, and 24.)

[139] Sebastian Thrun. An approach to learning mobile robot navigation. *Robotics and Autonomous Systems*, 15(4):301–319, 1995. doi: 10.1016/0921-8890(95)00022-8. (Cited on pages 37 and 107.)

[140] G. D. Tipaldi and K. O. Arras. Please do not disturb! minimum interference coverage for social robots. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1968–1973, Sept 2011. doi: 10.1109/IROS.2011.6094867. (Cited on page 37.)

[141] A Tversky and D Kahneman. The framing of decisions and the psychology of choice. *Science*, 211(4481):453–458, 1981. ISSN 0036-8075. doi: 10.1126/science.7455683. (Cited on page 6.)

[142] E. Ugur, M. R. Dogar, M. Cakmak, and E. Sahin. The learning and use of traversability affordance using range images on a mobile robot. In *2007 IEEE International Conference on Robotics and Automation*, pages 1721–1726, April 2007. doi: 10.1109/ROBOT.2007.363571. (Cited on pages xiii, 35, 36, and 61.)

[143] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2094–2100. AAAI Press, 2016. (Cited on pages 2 and 37.)

[144] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 3024–3030. AAAI Press, 2015. ISBN 0-262-51129-0. (Cited on pages 108, 109, 110, and 115.)

[145] Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth J. Teller. Learning semantic maps from natural language descriptions. In *Robotics: Science and Systems*, 2013. ISBN 978-981-07-3937-9. (Cited on pages 30 and 31.)

[146] C. Wang, K. V. Hindriks, and R. Babuska. Active learning of affordances for robot use of household objects. In *IEEE-RAS International Conference on Humanoid Robots*, pages 566–572, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041419. (Cited on page 39.)

[147] Elke U Weber and Osman Coskunoglu. Descriptive and prescriptive models of decision-making: Implications for the development of decision aids. *IEEE transactions on Systems, Man, and Cybernetics*, 20(2):310–317, 1990. doi: 10.1109/21.52542. (Cited on pages 7 and 35.)

[148] J. Wu, H. I. Christensen, and J. M. Rehg. Visual place categorization: Problem, dataset, and algorithm. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4763–4770, Oct 2009. doi: 10.1109/IROS.2009.5354164. (Cited on pages 30 and 31.)

[149] Hendrik Zender, O Martínez Mozos, Patric Jensfelt, G-JM Kruijff, and Wolfram Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008. doi: 10.1016/j.robot.2008.03.007. (Cited on pages xiii, 30, 31, and 43.)

[150] H. Zhang, C. Reardon, F. Han, and L. E. Parker. Srac: Self-reflective risk-aware artificial cognitive models for robot response to human activities. In *2016 IEEE International Conference on Robotics and Automation*, pages 3301–3308, May 2016. doi: 10.1109/ICRA.2016.7487503. (Cited on page 7.)

[151] Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010. (Cited on pages 7 and 66.)

[152] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI*, pages 1433–1438. AAAI Press, 2008. ISBN 978-1-57735-368-3. (Cited on page 66.)

[153] Kai Zimmermann and Christian Freksa. Qualitative spatial reasoning using orientation, distance, and path knowledge. *Applied Intelligence*, 6(1):49–58, 1996. ISSN 1573-7497. doi: 10.1007/BF00117601. (Cited on page 33.)

[154] V. A. Ziparo, L. Locchi, D. Nardi, R. R. Palamara, and H. Costelha. *Petri net plans: A formal model for representation and execution of multi-robot plans*, volume 1, pages 80–87. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2008. ISBN 9781605604701. (Cited on pages 21 and 22.)