SAPIENZA
UNIVERSITÀ DI ROMA

Spatial Representation for Planning and Executing Robot Behaviors in Complex Environments

Francesco Riccio

ID number 1572104

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

XXX Cycle

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

April 2018

Thesis Advisor
Prof. Daniele Nardi

Co-Advisor
Prof. Barbara Caputo

Thesis not yet defended

Spatial Representation for Planning and Executing Robot Behaviors in Complex Environments

Keywords: Robot Learning, Reinforcement Learning, Policy Improvement, Knowledge Representation, Decision Making, Spatial Representation

Ph.D. thesis. Sapienza University of Rome

Version: July 20, 2018

Website: https://www.diag.uniroma1.it/∼riccio

Author's email: riccio@diag.uniroma1.it

Abstract

Robots are already improving our well-being and productivity in different applications such as industry, health-care and indoor service applications. However, we are still far from developing (and releasing) a fully functional robotic agent that can autonomously *survive* in tasks that require *human-level* cognitive capabilities. Robotic systems on the market, in fact, are designed to address specific applications, and can only run pre-defined behaviors to robustly repeat few tasks (e.g., assembling objects parts, vacuum cleaning). They internal representation of the world is usually constrained to the task they are performing, and does not allows for generalization to other scenarios. Unfortunately, such a paradigm only apply to a very limited set of domains, where the environment can be assumed to be static, and its dynamics can be handled before deployment. Additionally, robots configured in this way will eventually fail if their "handcrafted" representation of the environment does not match the external world.

Hence, to enable more sophisticated cognitive skills, we investigate how to design robots to properly represent the environment and behave accordingly. To this end, we formalize a representation of the environment that enhances the robot spatial knowledge to explicitly include a representation of its own actions. Spatial knowledge constitutes the core of the robot understanding of the environment, however it is not sufficient to represent what the robot is capable to do in it. To overcome such a limitation, we formalize SK4R, a spatial knowledge representation for robots which enhances spatial knowledge with a novel and *functional* point of view that explicitly models robot actions. To this end, we exploit the concept of *affordances*, introduced to express opportunities (actions) that objects offer to an agent. To encode affordances within SK4R, we define the *affordance semantics* of actions that is used to annotate an environment, and to represent to which extent robot actions support goal-oriented behaviors.

We demonstrate the benefits of a functional representation of the environment in multiple robotic scenarios that traverse and contribute different research topics relating to: robot knowledge representations, social robotics, multi-robot systems and robot learning and planning. We show how a domain-specific representation, that explicitly encodes affordance semantics, provides the robot with a more concrete understanding of the environment and of the effects that its actions have on it. The goal of our work is to design an agent that will no longer execute an action, because of mere pre-defined routine, rather, it will execute an actions because it "knows" that the resulting state leads one step closer to success in its task.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Artificial agents that are able to autonomously decide *what* to do, *when* and *how* to do it, is one of the most fascinating and appealing challenges, that humanity is facing in this century. Researchers in the field of Robotics and Artificial Intelligence (AI) are pursuing such an ultimate goal by analyzing every aspect of human psychology, locomotion and cognitive capabilities in order to develop agents that are able to process information and reach human-level performance.

Robotics and AI have succeeded in multiple scenarios, improving our everyday well-being and productivity in industry [149], health-care [25] and house keeping [218]. However, in nowadays applications, robots usually rely upon "handcrafted knowledge" and only show pre-defined behaviors in order to reach a satisfactory compromise between performance and robustness [190]. We refer to handcrafted knowledge as concepts and facts explicitly modeled by an expert operator in the robotic platform. Such a type of knowledge tells the robot about specific concepts restricted to the task, and does not allow for reasoning beyond the task domain. The industrial scenario is an exemplar case where handcrafted knowledge is used. In this setting, robots perfectly repeat a restricted set of tasks, without any "understanding" of what they are doing. Such a paradigm can only be applied in controlled environments and very limited domains, where it is possible to foresee every event and operate with pre-defined routines. Despite the optimal and repeatable behaviors, this paradigm cannot be adopted on robots deployed in uncontrolled and dynamic environments [154] (e.g., shared workspace with human beings). Unfortunately, generating human level cognitive capabilities is not trivial due to an uncountable amount of issues spanning from perception [231] to natural language understanding [148, 221]. Agents must be able to represent information in a reusable way, in order to allow reasoning at any point in time [46]. Humans are able to reuse previous information and overcome new problems, learn from personal experience, learn from natural language dialogues, react spontaneously to unexpected stimuli, and reason about others' intentions and actions. All these skills are possible thanks to the ability of representing the state of the external world, store meaningful concepts, and generalize this continuous flow of information to new situations.

In this thesis, we aim at pushing the current state-of-the-art in cognitive robotics towards these skills by introducing (1) practical approaches to represent environmental knowledge [155] and (2) novel techniques for decision making and behavior

generation [133]. Our goal, in fact, is to construct a representation of both *environmental* and *task-related* knowledge, which can be easily used to shape robot behaviors. To this end, we introduce SK4R a spatial knowledge representation for robots, that is designed to encode high-level concepts and support behavior execution. Throughout this thesis, we present the SK4R architecture and show how it can be used to express spatial knowledge on multiple robotic platforms in various applications. Our aim is to demonstrate that semantic spatial knowledge is key to improve performance in several scenarios where a robots cannot feature only "automated" behaviors. Hence, the focus of this thesis consists in demonstrating that

Spatial knowledge representation is an important feature that robots must have in order to properly perceive and act in the external world.

To this end, we motivate our approach and highlight the contributions to the current state-of-the-art of this thesis work. Spatial knowledge in a mobile robot enables and facilitates successful planning and execution of actions in complex environments. We specifically focus on scenarios involving large-scale, dynamic, human environments, such as office buildings and homes [158]. We assume that a mobile robot is physically capable of sensing the environment using on-board sensors. This requirement is fundamental for cognitive robots that need to observe the environment, guarantee autonomous behaviors and update their beliefs while operating [36].

1.1. Motivations

We motivate our work by (1) discussing the importance of spatial knowledge (with an example); (2) describing the role that SK4R is expected to play in order to properly shape robot behaviors; and (3) describing the desired features that a spatial knowledge representation should have.

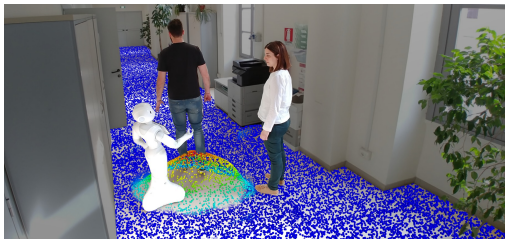
1.1.1. Motivating Example

Let us imagine the scenario of a robot with the task to follow a person [92, 171], as pictured in Figure 1.1. While it may seem rather simple, in order to succeed, the robot has to account for several environmental features, task-related information and handle dynamic events. In fact, our actions and position change in accordance with social relationships (e.g. work hierarchies, parenting, a peer), the number of people we are following, the location of the task (e.g. corridor, open areas), dynamic events (e.g. people passing by, obstacles, other robots), time constraints, other people intents [21]. For a robot, this task is even more difficult due to partial observability and noise in both the perception and actuation system.

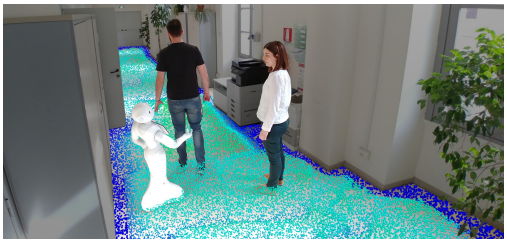
To complete this task, the robot cannot just reason about geometric measurements and raw sensor data, but it has to abstract its observations and assign to them a semantic meaning. This allows to relate raw sensor readings to concepts that characterize the state of the world in a more abstract sense. For example, in order to support effective decisions, the robot has to feature the possibility to reason

Figure 1.1. Scenario of a following task. The robot has to maintain a desired distance to the target while reacting to external stimuli.

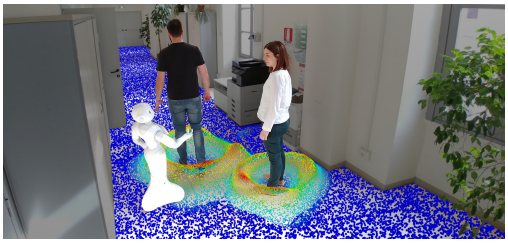
about relative positions of a group of people, rather than pixel values retrieved directly from the camera sensor. This is usually achieved through a domain-specific representation of the environment that offers the ability to abstract observations to high-level concepts – explicitly formulated to support robot decision. The goal of our representation SK4R is indeed, to provide such a level of understanding of the environment and let the robot operate in it by reasoning about high-level concepts, and the effects that its actions have on the environment itself. A possible solution, to generate a high-level description of the world state, can be achieved by modeling relevant features individually and merge them altogether within a unique representation, such as a gridcell map. For example, we can (1) localize people [24], locate doors and obstacles [7], determine safety distances [132], and detect objects [167]; and then, (2) use these processed information to annotate areas of the environment accordingly. Figure 1.2 depicts a possible output of this solution, where the space is characterized as a gridcell map, and each cell has a score indicating, to which extent, a given area satisfies the task. The output is a probability distribution, not only of navigable spaces, but also of areas of the environment that satisfy the different constraints imposed by the task: position of the person to follow with respect to the arriving point (Figure 1.2a), obstacles and objects in the scene (Figure 1.2b) and safety constraints (Figure 1.2c). Also, such a solution can be used to account for spatial constraints not directly perceivable by the robot such as preferred side of a corridor, inter-agent proxemics settings and people intentions [21] (Figure 1.2d). Finally, Figure 1.2e shows the spatial distribution satisfying all the high-level constraints and that allows the agent to correctly interpret the environment with respect to its task objective. It is important to remarking that without a spatial representation encoding these high-level constraints, the robot cannot fulfill the task requirements and its behavior may result not proper and ineffective. Such a solution is beneficial for two reasons: firstly it allows to semantic annotation of the space, and secondly, it provides an explicit representation of how the robot should move within the environment and select its actions. In contrast, existing spatial representations do not explicitly relate robot actions to the environment, rather, they address the problem of knowledge representation and planning separately [232]. In this thesis,



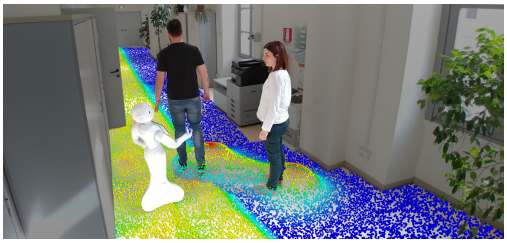
(a) Target and Destination



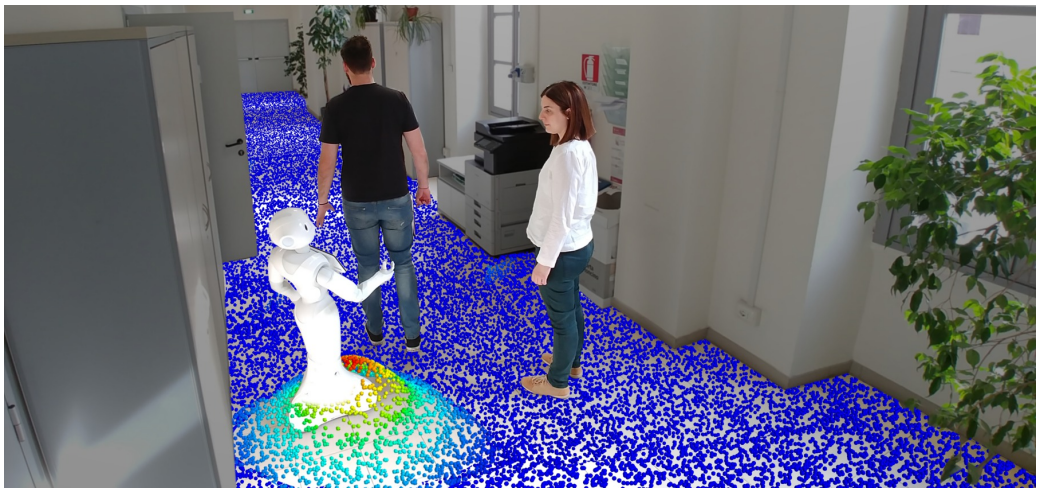
(b) Obstacles and objects



(c) Safety



(d) Social rules



(e) Affordance semantics of the following task

Figure 1.2. Spatial constraints highlighted to support a correct execution of a following task. To succeed, the robot has to consider the position of the person to follow and the arriving point (top-left), obstacles and objects (top-left) and safety constraints (middle-left). Also, service robots have to consider social rules such as preferred side of a corridor and inter-agents proxemics settings (middle-right). At the bottom the affordance semantics of the following task that is used to satisfy all the high-level constraints.

we demonstrate that explicitly modeling robot actions and spatial knowledge within the same representation is crucial to enable cognitive robots capabilities. In fact, we define SK4R as a functional representation of the environment that can be implemented and used to annotate the environment, and support decision making.

1.1.2. Role of a Spatial Knowledge Representation

Referring to the discussion of roles of a knowledge representation in [45] – where the authors analyze different requirements and features that a representation should have, and a more specific analysis for spatial knowledge in [157], we expect SK4R to assume a set of roles for an autonomous robot:

- a substitution (surrogate) of the world that allows the robot to reason about actions involving parts of the environment beyond its sensory horizon. The surrogate is intended to represent the belief [91, 201] about the state of the world and express what the robot can do to accomplish its task. It is a way of structuring spatial information so that it is computationally feasible to perform inferences and action planning. It is important to note that such a representation is inherently imperfect, i.e. it is incomplete (some aspects of the world are not represented), inaccurate (captured with uncertainty), and likely to become invalid (e.g. due to the dynamics of the world);
- a definition of relevant aspects of the world, and a specification of the formalism used to represent and relate them. To this end, the representation defines the levels of abstraction at which spatial entities (e.g. objects, obstacles, area topology) exist and the types of relations among them. It is worth noticing that this significantly affects the ability of the robot to plan and execute specific actions. Furthermore, the representation does not require to be more expressive than needed by the robot in order to successfully act. It *only* has to represent aspects of the world relevant to characterize the task of the robot, and to support its actions;
- a medium of communication between the robot and humans. In scenarios involving human-robot collaboration, spatial knowledge provides a common ground for communication and knowledge transfer. The representation must therefore be capable of connecting human spatial concepts to those internal to the robot.

1.1.3. Desired Properties of the Representation

Having in mind the roles of a representation, practical limitations, and existing robotic systems [214, 116, 132, 79], we now describe desired properties of a spatial knowledge representation in realistic dynamic environments. Given the robot sensing, it is useless to represent the environment as accurately as possible. A very accurate representation is likely to be intractable and requires a substantial effort to be kept up-to-date. Moreover, its usability remains constrained by robot capabilities. Hence, the primary property is that the representation should instead be minimal and the spatial knowledge should be represented only as accurately as required to support the functionality of the robot.

Planning is a computationally demanding process and its complexity increases exponentially with the size of the environment and number of considered spatial entities. However, due to the way real-world environments are structured and limitations of robot sensors and actuators, decomposing the planning problem can greatly reduce its complexity while maintaining highly optimal results. The representation, in fact,

has to be able to provide support for both long-term global plans and short-term local behaviors [132, 11, 79]. To this end, a spatial representation should perform knowledge abstraction, characterizing spatial phenomena of gradually increasing complexity. This leads to an intuitive representation of the continuous space, which significantly reduces the number of states for planning [81] and provides a basis for higher-level conceptualization [232].

Representing uncertainty in the belief state is crucial for the robot to make informed decisions in the real-world, including planning for epistemic actions and anticipating future uncertainty. In this context, decision-theoretic planning algorithms rely on probabilistic representations of uncertainty, therefore, it is desirable for a knowledge representation to also have a probabilistic formulation.

Once the spatial representation is formalized, our goal is to design a robotic agent able to exploit such a representation and deliberate decisions. The agent, in fact, must be able to acquire new information, update its beliefs and use such a knowledge to leverage its actions. We consider such a capability to connect perception to actions as the central focus of our dissertation, and to be typical of "intelligent robots".

1.1.4. Desired Robot Capabilities

In this section, we sketch how the behavior of an agent is influenced by the type of knowledge that it owns [74, 196]. The relation between knowledge representation and actions, is indeed the one that we pursue in this thesis, and that we investigate through SK4R. Generally, while robotics mainly focuses on motions and perceptions, the mission of AI is to interpret observations and map them to actions [188], in order to generate goal-oriented behaviors [11]. In this thesis, we adopt this view and aim at enabling intelligent behaviors in robotics. Our goal is to provide robots with the ability to semantically interpret the environment, and with specific abilities that are needed to overcome everyday challenges and scenarios. We focus on robots that are expected to handle situations with *human-level* skills and interpret the world in a *human-compatible* manner. Motivated by previous work in the field [46, 56, 232, 93], we present different skills that we want to analyze and embed in intelligent agents. A *spatial-knowledge aware* robot should be able to:

- acquire knowledge during its operation. It is not possible to assume that the agent can be pre-programmed with all the knowledge that it will need during its life-time. Hence, instead of attempting to encode this general information, we have to endow robots with the ability to acquire new knowledge as required;

- reason about experience. This ability is also referred to as "reflection" [22, 131]. It is defined as the ability of the agent to self organize its past experience and generalize from it to new situations. This is an important skill that enables the reuse of knowledge and forces the agent to organize it effectively;

- feature multi-modal perception schemes. There is not a single source of knowledge and we need to enable robots to exploit everything that their sensors can perceive. A intelligent agent, in fact, has to be able to fuse information coming from different sources in order to generate a complete awareness of its surroundings;

- anticipate and interpret others actions and intentions in order to cooperate and coexist. Hence, when planning for its actions, the robot has to both take into account its objectives and consider other agents;
- interpret the environment in a human compatible manner. This to enable grounded interactions with humans and other non-homogeneous agents. It needs to represent the environment and reason at an higher level of abstraction that allows semantic representation of entities and their relation.

Our aim is to enable effective and practical capabilities through spatial knowledge and to show that SK4R, indeed, improves the performance of a robotic system in task execution. Our goal is to contribute in enabling robots to behave in complex environments and achieve a *human-level* understanding about the environment and task related information.

1.2. Contributions

This section describes the main achievements of this thesis and highlights its specific research contributions:

1. a representation of spatial knowledge designed and implemented to support decision making for robots. The representation allows the robot to interpret the environment at different levels of abstraction, thus enabling it to focus on important aspects of the world and to perform decision-making. Independently of the scenario of deployment, SK4R allows the robot to deliberate an action plan by relying on the chosen model to represent the environment;
2. dynamic knowledge acquisition and update based on sensory information. The SK4R representation enables the robot to gather new knowledge and integrate it into the current belief of the agent. This is an important feature which is usually achieved by updating a world model both in the case of *explicit* representations (e.g. grid world of the 2D space, semantic maps) and *implicit* representations (e.g. neural networks)[1];
3. in social robotics, robots have to interpret the social context in which they operate. We investigate how SK4R$_E$ can be used to represent such a context, and which features of the environment are important to represent in order to socially interact with humans. To this extent, we frame our contribution within the paradigm of symbiotic autonomy [183]. We enable SK4R to represent environmental factors that favor human-robot interactions and contribute to improve a more natural coexistence.
4. SK4R is a general representation that can be implemented in different robotic applications. We show-case its effectiveness in different robot coordination/cooperation task. Multi-robot scenarios are one of the most challenging

[1]Explicit representations describe the environment through a model which exposes spatial entities and their features. Implicit representations, conversely, update an internal belief of the environment which is usually not accessed by external users.

research area in robotics, where behavior learning and decision-making approaches are often too specific and overfit the application of deployment. In this setting, we contribute to a generalization of SK4R to the multi-robot scenario, which enables distributed robot control. We propose a novel representation that allows for distributed belief update, world modeling and task assignment in adversarial and cooperative scenarios [179, 175];

5. SK4R enables practical robotic planning applications by attacking the "curse-of-dimensionality" with a combination of Monte Carlo Tree Search (MCTS) methods and Q-learning. The curse-of-dimensionality, in fact, is a hard problem that needs to be tackled every time a decision theoretic algorithm is employed. Such a problem is contributed by several factors that heavily influence the complexity of used solutions. Those factors – such as the number of environmental features (landmarks), their dimensionality, the complexity of the problem and the structure of the environment – exponentially increase the dimension of the state-space. To alleviate this problem, SK4R enables robots to learn directly from the interaction with the environment, as well as to implicitly represent, and act in it [175, 177]. Our representation allows for focused exploration of the search space and permits the robot to (1) iteratively improve its action policy, and (2) generalize to unseen states of the environment;

6. we extend the formalization of SK4R to perform hierarchical optimistic planning to address task decomposition, and further alleviate the "curse-of-dimensionality". To this extent, our contribution borrows the concept of Hierarchical Task Networks (HTNs) [60] to generate a hierarchy of robot actions that is used to (1) guide exploration during policy refinement [35], use reflection [131], (3) and improve robot performance in addressing composite tasks.

7. SK4R contributes to enable deep learning in robotics applications. Deep learning shows remarkable results in the field of reinforcement learning with virtual agents [134]. However, since it requires a significant number of training samples, it remains challenging in robotics. SK4R contributes in this direction by introducing a novel technique that combines policy improvement and DNN-based function approximators in order to allow for generalization and learning of competitive policies with (i) few training samples and (ii) few iterations of the learning algorithmic;

1.3. Thesis Organization and Publications

The thesis is organized in three main parts: the preliminary Part I which introduces the theoretic background, and the related work upon which we build our research. Part II which describes SK4R as well as its components and applications, and the final Part III which summarizes the thesis, discusses it and points toward new research directions. Each part is then organized as follows

1.3.1. Part I: Preliminaries

Chapter 2: Background

The background chapter introduces and describes the basic theoretic concepts and notions that we rely on throughout this work. The chapter illustrates basic concepts about knowledge representation and reasoning, with a particular focus on *Markov Decision Processes*. Then, it covers the main notions of robot learning and planning, in the context of reinforcement learning. Finally, the chapter introduces building the elements of convolutional neural networks and deep learning.

Chapter 3: Related Work

This chapter surveys current state-of-the-art approaches that relate to this thesis. It presents and categorizes current approaches to spatial knowledge representation both for single and multi-robot applications. Accordingly, it describes reasoning and planning methods that exploit such spatial representations. The chapter introduces state-of-the-art techniques in robot learning and planning, by analyzing their strength and limitations. Then, it describes recent advances in deep robot learning and discusses their applicability to robotic scenarios. The chapter details the relation and connection of the reported literature to SK4R. Part of the approaches reported in this chapter have been analyzed in published survey papers

- Riccio, F., Lázaro, M.T., Gemignani, G. and Nardi, D., 2015. *Multi Robot Perception and Action: World Modeling and Task Allocation.* In RSS Workshop on Principle of multirobot systems ([178]).

- Bloisi, D.D., Nardi, D., Riccio, F. and Trapani, F., 2016. *Context in robotics and information fusion.* In Context-Enhanced Information Fusion (pp. 675-699), Springer ([19]).

1.3.2. Part II: Spatial Knowledge for Robots

Chapter 4: SK4R: Spatial Knowledge for Robots

This aim of this chapter is to introduce SK4R and its components. Its core part is the description of the adopted spatial representation and the implementation in an exemplar robotic scenario. The chapter describes the general SK4R representation along with an evaluation on a people *following* task. Published papers related to this chapter are

- Riccio, F., Capobianco, R., Hanheide, M. and Nardi, D., 2016, June. *STAM: A framework for spatio-temporal affordance maps.* In International Workshop on Modelling and Simulation for Autonomous Systems (pp. 271-280), Springer ([171]).

- Riccio, F., Capobianco, R., Nardi, D., *Using Spatio-Temporal Affordances to Represent Robot Action Semantics*, In Machine Learning Methods for High-Level Cognitive Capabilities in Robotics Workshop@IROS 2016 ([179]).

Chapter 5: SK4R for Indoor Robots

This chapter presents the result of implementing SK4R over an explicit hierarchical representation. It details the implementation of SK4R in a indoor office environment by analyzing and evaluating the benefits of SK4R in this complex scenario. Published papers related to this chapter are

- Pronobis, A., Riccio, F. and Rao, R.P., 2017, June. *Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments*. In ICAPS 2017 Workshop on Planning and Robotics, Pittsburgh, PA, USA ([158]). The paper has been also presented at the RSS 2017 Workshop on Spatial-Semantic Representations in Robotics, Boston, MA, USA ([159]).

Chapter 6: SK4R for Social Interactions

This chapter presents the results of a user-study where SK4R is used to represent *social* environmental features. The goal of this contribution is to highlight social factors that may, or may, not influence social interactions. Published papers related to this chapter are

- Riccio, F., Vanzo, A., Mirabella, V., Catarci, T. and Nardi, D., *Enabling Symbiotic Autonomy in Short-Term Interactions: A User Study*, In Social Robotics - 8th International Conference, ICSR 2016, Kansas City, MO, USA, November 1-3, 2016, Proceedings, Springer International Publishing, vol. 9979, Kansas City, MO, USA, pp. 796-807, 2016 ([174]).

- Vanzo, A., Riccio, F., Sharf, M., Mirabella, V., Catarci, T. and Nardi, D., Who is Willing to Help Robots? A User Study on Collaboration Attitude, *journal paper submitted*, 2018 ([1]).

- Capobianco, R., Gemignani, G., Iocchi, L., Nardi, D., Riccio, F. and Vanzo, A., 2016, March. *Contexts for Symbiotic Autonomy: Semantic Mapping, Task Teaching and Social Robotics*. In AAAI Workshop: Symbiotic Cognitive Systems ([34]).

Chapter 7: SK4R for Multi-Robot Search

In this chapter, we show the implementation of SK4R in very challenging and complex scenarios, where a team of robots is involved in a active search for an object. The chapter describes the proposed approaches and reports the results and benefits of using SK4R to improve the overall performance of a team of robots in both (1) a cooperative and adversarial setting and (2) a coordination task in an office environment. Published papers related to this chapter are

- Riccio, F., Borzi, E., Gemignani, G. and Nardi, D., 2015, July. *Context-based coordination for a multi-robot soccer team*. In Robot Soccer World Cup (pp. 276-289), RoboCup Symposium 2015, Springer ([169]).

- Riccio, F., Borzi, E., Gemignani, G. and Nardi, D., 2016, October. *Multi-robot search for a moving target: Integrating world modeling, task assignment and context.* In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on (pp. 1879-1886), IEEE ([170]).

Chapter 8: SK4R for Optimistic Planning

This chapter provides a detailed description of a novel approach for robot planning in the context of reinforcement learning. The chapter discusses the contribution to the current state-of-the-art and reports an extensive experimental evaluation in different robotic tasks and environments. Published papers related to this chapter are

- Riccio, F., Capobianco, R. and Nardi, D., 2016, June. *Using monte carlo search with data aggregation to improve robot soccer policies.* In Robot World Cup (pp. 256-267), RoboCup Symposium 2016, Springer ([173]).

- Riccio, F., Capobianco, R. and Nardi, D., 2016, November. *Learning human-robot handovers through π-STAM: Policy improvement with spatio-temporal affordance maps.* In Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on (pp. 857-863), IEEE ([172]).

- Riccio, F., Capobianco, R., Nardi, D. *Q-CP: Learning Action Values for Cooperative Planning*, In proceedings of 2018 IEEE International Conference on Robotics and Automation (ICRA) 2018 ([175]).

- Riccio, F., Capobianco, R., Nardi, D., DOP*: Deep Optimistic Planning with Approximate Value Function Evaluation*, In proceedings of the 2018 International Conference on Autonomous Agents and Multiagent Systems (AAMAS) 2018 ([177]).

- Riccio, F., Capobianco, R., Nardi, D., LoOP*: Iterative Learning for Optimistic Planning on Robots*, submitted to journal paper, ([176]).

Chapter 9: SK4R for Hierarchical Optimistic Planning

This chapter extends the formulation presented in Chapter 8 to the case of hierarchical optimistic planning. It describes the building concepts of the proposed algorithm and reports an experimental evaluation on two robotic applications. Published papers related to this chapter are

- Capobianco, R., Riccio, F., Nardi, D., *Hi-Val: Iterative Learning of Hierarchical Value Functions for Policy Generation*, In proceedings of 2018 fifteenth International Conference on Intelligent Autonomous Systems IAS-15, ([35]).

1.3.3. Part III: Conclusion

Chapter 10: Conclusions

This chapter concludes the thesis by summarizing it and recalling its main contributions and achievements. It then discusses the thesis work and criticize its current limitations to point out towards new research directions.

Part I

Preliminaries

Chapter 2

Background

Different research studies, in both the fields of Robotics and AI, contribute to the background and basic notions of our dissertation. We recall common techniques to represent and encode knowledge into artificial agents, as well as methodologies and techniques to specifically encode spatial knowledge into robotic systems. The chapter then describes models of reasoning by focusing on Markov Decision Processes [163] both for single and multi-agent configuration. Finally, we recall elements of robot reinforcement learning, Monte-Carlo Tree Search planning and deep learning.

2.1. Knowledge for Robots

It is important to understand which aspects of the world we want to represent, and what kind of knowledge the agent has to consider in order to efficiently act [13]. Autonomous robots should be aware of three major aspects, that enable reasoning about (i) the external world of operation, (ii) mission and task-related information (iii) and internal status of the agent itself. To better understand these three aspects of knowledge, let us consider the scenario in which a team of robots has to complete a *coverage* task. This is a widely used task in robotics [80], where a team of robots has to visit every area of a scenario, and update its belief about the environment. The coverage task can be implemented to address various applications such as surveillance [195], search [170] and house-cleaning [218]. To describe a coverage task, let $\boldsymbol{r} = \{r_j\}_{j=0}^{R}$ be the set of R robots, and $\boldsymbol{g} = \{c_g\}_{g=0}^{G}$ is the set of G cells in which the environment has been discretized; Figure 2.1 illustrates the robots (red), their field of view (yellow), the world representation (black lattice) and the current status of the task (the gray color represents areas still to be visited). Each cell c_g can be represented by a binary variable being 1 if it has been visited (by at least one robot), and 0 otherwise. In a basic implementation of the coverage problem, each cell needs to be visited once, and both dynamics of the world and robot actions are deterministic.

Example

In this setting, we can highlight the types of knowledge that the team has to consider to succeed. The teams needs to interpret the environment, detect relevant features and understand their relations. For example, if the robots are tasked to cover particular areas, (e.g. bedrooms) then they have to know how to characterize those areas and their constituting elements (e.g. beds, wardrobe). In other words,

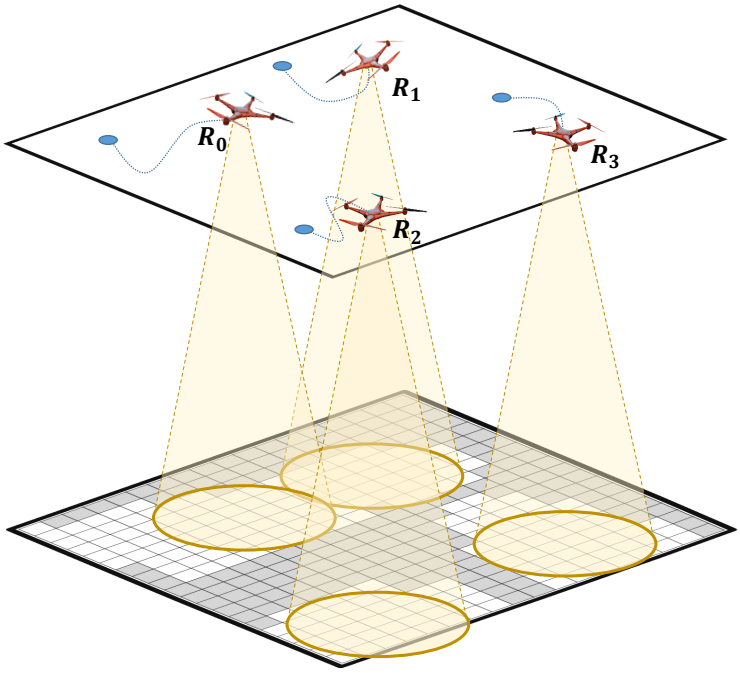


Figure 2.1. Coverage task with a team of aerial robots. The robots are represented as red dots and their field of view is in yellow. The scenario has been discretized with a grid-map (black lattice) and the status of the coverage task is highlighted in gray for cells not visited yet, while white visited ones.

the robots need to know how to represent important features of the environment and how to relate them.

Moreover, in order to operate efficiently, the robots need to share and update the current status of the task: which areas are not visited yet; which agent is assigned to cover an area; which are the operational constraints and requirements of the mission (e.g. timing, number of agents, communication overload). This is key to formalize information related to the ongoing task, and to guarantees an effective and correct advancement towards its completion.

Each robot has to be able to analyze itself and understand how to behave in accordance with its internal state (e.g. battery level, malfunctioning). For example, if a robot is running out of charge, it has to notify the rest of the team. In this way, the teammates can reassemble their formation and adapt to the new scenario.

These three knowledge perspectives are crucially important for an autonomous robot and have to considered to enable effective and efficient behaviors. In fact, this is a well-known categorization of knowledge, introduced in the late '90s by Turner [219], which describes *contextual knowledge* as the sum of *environmental*, *task-related* and *self*-knowledge:

Contextual knowledge

- **Environmental knowledge.** This kind of contextual information formalizes data that is environment-dependent and that does not directly depend on the robot actions. The robot perceives the world through its sensors and it infers the current status of the scenario (e.g., presence of obstacles or people). In a navigation system, for example, the robot can tune its parameters depending

on the terrain conditions. Equivalently, the information about the illumination conditions can be used to improve the perception or to discern the saliency of information as related to the task. In the case of a coordinated team of robots – such as unmanned aerial vehicles (UAVs) [195] – performing a coverage task, the robots may adapt their navigation parameters according to the detected conditions of the environment (e.g., terrain type).

- **Task-related knowledge.** Task-related information is generally defined by the mission specifications. Depending on the operating conditions and on the task constraints (e.g., time constraints, priorities, and locations), the robot adapts its execution plan in order to increase robustness and efficiency. It is worth noting that the knowledge about a task does not modify its outcome or requirements, but it is exploited to influence the execution of the task with the aim of improving the performance. Using the previous example, the team of robots can accomplish its mission in different modalities by considering: (i) the current day time; (ii) the location to cover; (iii) the information processed by teammates; (iv) additional information gathered during the mission (e.g., paused teammates or already visited areas).

- **Self-knowledge.** In this case the robot infers context knowledge by relying on its own status and on the internal representation of the surrounding environment. This type of knowledge is fundamental to provide the robot with the ability of self analyzing its performance, and dyagnose its components. In our example, one of the teammates could self recognize a malfunctioning and communicate its status to the team. Consequently, the team can consider unreliable the information coming from that particular robot. It is worth noting that, discarding information coming from a malfunctioning robot can prevent the system from poor performance, or even to fail in completing the task (e.g. areas not covered by any teammate).

In accordance with this taxonomy, *knowledge* can be categorized in three classes that cover different aspects required by a cognitive robot to demonstrate "awareness" about *environmental*, *task-related* and *self-* knowledge. Once contextual knowledge is gathered, a common representation is usually employed to reason about the collected knowledge. A *context representation* has to provide a uniform view of the collected data and a robust reasoning process for state estimations, behaviors specialization, and task execution evaluations. We analyze existing approaches by emphasizing the differences between representation criteria by grouping them into three classes: *Embedded*, *Logic*, *Probabilistic*.

Context representation

Systems using *Embedded Context Representation* represent context as sets of meaningful sensory features that characterize particular situations and trigger particular routines of the system. Context classification with different sets of features is used for robots relying on visual perception such as scouting robots, and more generally, on systems performing visual recognition. Narayanan et al. [138] model reactive behaviors for a mobile robot according to a set of scenarios – each of which, consists of traces of visual frames with the respective desired movements. During the execution of its tasks, the robot scans the environment and tries to build correlations between the sensed world and the demonstrated scenario frames. Once a correlation

Embedded

is established, the current situation is identified and the robot actuators execute the requested motion law. When image classification (or scene recognition) techniques are involved, *a priori* knowledge about the geometrical and visual properties of known classes of objects can be gathered and used to direct the recognition process more efficiently [123]. These properties can be encoded explicitly as desired values for functions representing particular visual features; or implicitly, as collections of frames displaying the desired features. Similarly, Buch et al. [27] and Costante et al. [44] exploit specific features in the image reference frame to evaluate the alignment pose between objects in an image [27], and to define a visual classifier that clusters a target image with a *normalized-cut* based approach [44]. In order to increase the efficiency of the system, a measure of similarity with respect to the other previously labeled sets of images is computed before the classification step. Here, contextual information is represented as a set of labeled images, without any further abstraction about the classes they symbolize.

In some of these systems, causal relationship between current context and consequent behavior is modeled with rules in simple declarative formalisms. In these cases, the classification is performed through inference rules. These rules mostly represent fixed geometrical or physical constraints, and are not expanded by any cognitive learning process. Therefore, this solution can be assumed as a compromise between strategies based on symbolic knowledge approaches and the methodologies based on raw sensory data [124].

Logic-based

The most common choice in modeling contextual information is the use of *declarative knowledge representation languages*. Logic-based representations range from rule-based ontologies to first order logic. The main advantage in using such a representation is that a symbolic framework implicitly provides *inference tools*, which supports planning and reasoning. In Laird and Mohan [118] cognitive architectures integrate sensory data and descriptive knowledge with contextual rules, directly into the decision making process of the system. The decision procedure aims at modeling the current symbolic knowledge of the system, named *symbolic working memory*. The Symbolic Working Memory communicates to the perception layer and to the *permanent memory* modules, and it provides relational representations of recent sensory data, current goals, and long term memories. Contextual information is structurally defined within the permanent memory modules.

The challenging problem for this type of architectures is to develop context modules able to dynamically update and increment their knowledge. Indeed, turning experience into structured logical assertions requires a high level of abstraction, which is often difficult to achieve. Logic-based models need an accurate grounding of semantics into the sensed world. Karapinar and Sariel [99] describe a learning method for expanding and correcting contextual knowledge in robotics systems. The authors represent knowledge by means of *linear temporal logic* formulas, which allow the system to analyze episodes of failures occurred in past experiences and to adjust its internal knowledge. Whenever a failure occurs, the system identifies the related configuration of *risks of failures*. Therefore, the system learns how to connect possible failures to a *risk of failure* scenario, which can anticipate the failure itself. A system based on *formal representation languages* can be easily understood by human operators, which is the main advantage. Nonetheless, these techniques,

require a high level of abstraction of raw observations. Scalmato et al. [191] exploit human users to easy this task and provide a system with situation awareness. The use the human input to represent *Concepts* in the form of T-Boxes, while *contingent knowledge* (A-Boxes) is gathered during operation. This kind of representation is highly flexible, since a knowledge base building upon representation languages does not depend neither on the internal structure of the system nor on its domain. Therefore, it can be shared and adapted in different systems.

Probabilistic-based

A robotic system is affected at all levels by uncertainty in the agent observation and action execution. Hence, several approaches exploit probabilistic-based representations (e.g. Bayesian Networks) to address non determinism and dynamic settings. Witzig et al. [230] describe a collaborative human-robot system that provides contextual knowledge to enable more effective robotic manipulation. Contexts are represented by probability density functions, covering task specifications, known object properties or manipulator poses. The contextual knowledge is then used to assess the internal Bayesian Network, in order to model the grasp poses of the manipulator. Differently, Held et al. [82] exploit a probabilistic representation for object classification. To recognize other cars on the roadway, they allow the system to estimate the likelihood of membership of a particular element with respect to each category present in the learning process.

Knowledge representation methods strongly influence the architecture of the robotic system, and more in particular, reasoning and execution of actions. Indeed, each approach has its own strengths and weaknesses and multiple approaches can be combined to improve performance. Logic and probabilistic representations both supply effective structures for describing characteristics of environments, the former focusing on the expressiveness of the language and the latter on the reliability of the estimates. However, logic representations alone are not efficient to scale to huge state-spaces. On the other hand, a probabilistic encoding lacks descriptive power for modeling complex environments. Embedded representations, instead, rely on sub-symbolic structures for an effective mapping between the sensory data in input and the estimates for the contextual variables, but do not represent knowledge in a easily interpretable manner.

The research challenge objective in developing knowledge-based systems is, in fact, to design a framework and a suitable representation able to cover all the benefits that proposed approaches carry. Accordingly, our goal is to generate a knowledge representation able to provide a robotic platform with meaningful information and support action execution. To this end, in robotic settings, spatial knowledge is mandatory in order to represent and store world observations and reason about them. Different approaches exists to represent spatial knowledge at different levels of abstractions, each of which, focused on formalizing particular aspects of the environments.

In fact, for an autonomous robot, spatial knowledge is the core of a knowledge representation and its correct formalization is the primary problem to be tackled in order to enable robots to behave. In fact, depending on its formalization, spatial knowledge can encode (1) environmental features [70]; (2) task-related information [79]; and (3) self-awareness [158]. In line with this perspective, throughout our

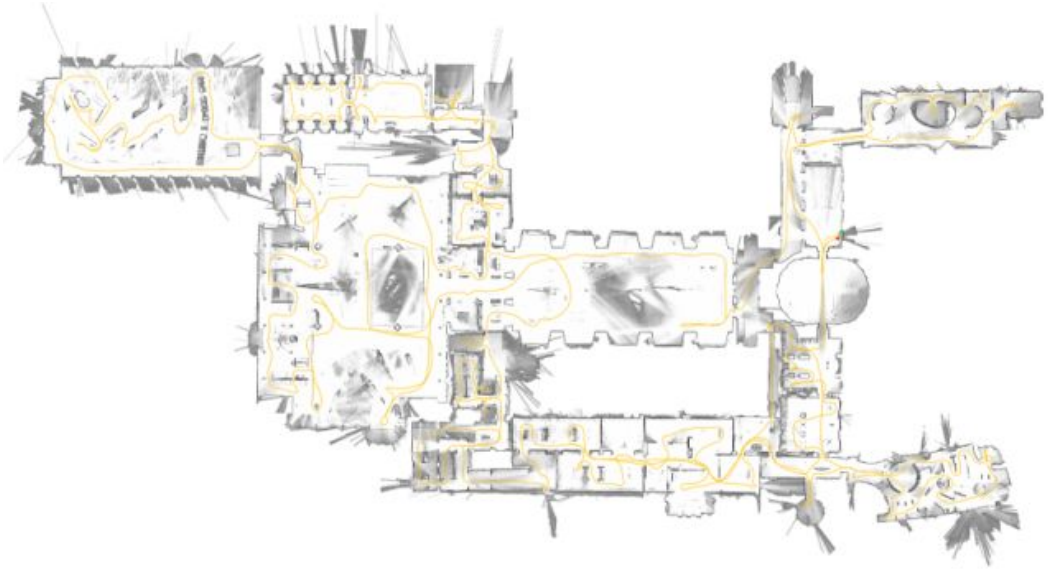


Figure 2.2. 2D metric map of the Deutsches Museum.

work we aim at designing a spatial knowledge representation that details meaningful aspects of the world which are key to support task execution.

2.2. Spatial Knowledge Representation

Representing the environment as a proxy to encode task-related information and self-knowledge, is a key issue to develop autonomous robots. In fact, in order to enable effective planning, the model of the environment is key to reason at different levels of abstraction: from long-term (more abstract) plans to short-term (more reactive) local behaviors. Existing techniques represent different aspects of spatial knowledge that can be arranged over three layers of spatial abstraction: metric, topologic and semantic.

Metric layer

Within the metric layer, the objective is to represent the environment by means of its geometry and accurate measurements. The first effort towards autonomy was centered in enabling the agent to represent the space of the environment by annotating *navigable* areas. These are usually modeled as grid-maps [76] that are obtained by relying on the on-board sensors of the robotic platform. A grid-map is a flexible representation adopted to model the environment as accurate as needed. Most importantly, it can represent any environment without any assumption on its structure (see Figure 2.2). More recently, the research in this areas, is focusing on more complex and accurate representations that allow to model the 3D nature of the environment. The most common approach is to represent the space as a dense cloud of points (Figure 2.3) that are able to capture every geometrical aspect of the environment [193].

These approaches, however, bound the spatial knowledge to geometrical features and mainly focus on representing the environment as accurately as possible. Nonetheless, reasoning at this layer of abstraction is difficult and computationally

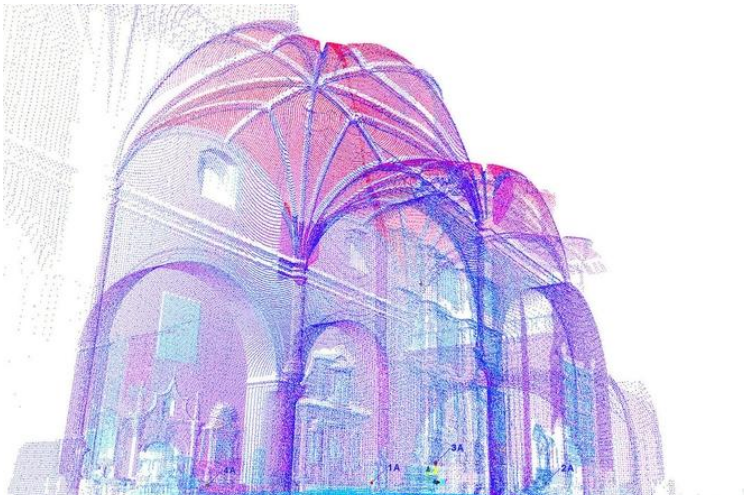


Figure 2.3. 3D point cloud of a church.

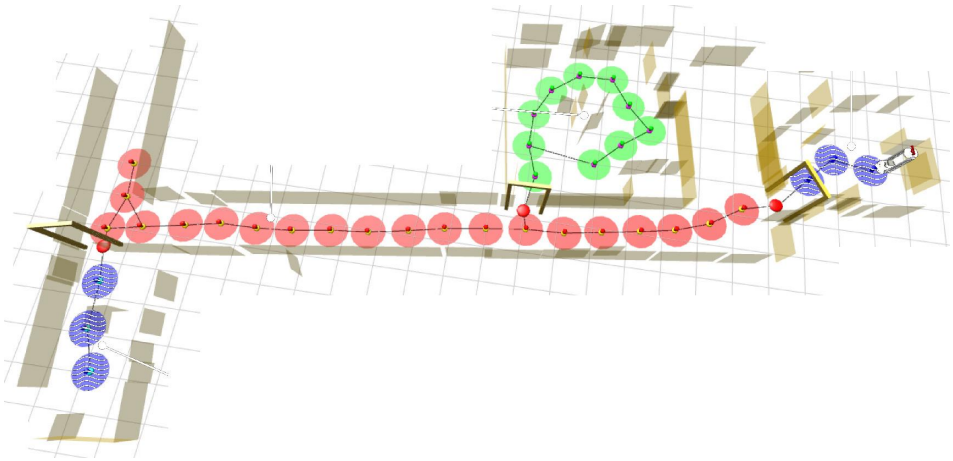


Figure 2.4. Topological map of an indoor office environments. The graph represents the topology of the building while node colors different areas (image from [161]).

expensive. Unfortunately, these representations do not provide any structure to support decision making. For this reasons, spatial knowledge has been enhanced to better understand the general topology and structure of the environment – and express it in a compact manner. This is generally referred to as a topological map of the space. Topological maps allow to represent the connectivity of areas and their relations (i.e. space topology). Generally, such a representation is built on top of the metric layer in order to ground areas to sensory observations, Figure 2.4. Given a metric representation of the environment, a node of the topological map can be added according to preferred criteria – usually navigability and relevance – and edges can be added to highlight traversability between nodes [158]. Such a layer of abstraction encodes sufficient information to accomplish different tasks. For instance, nodes can be placed to highlight objects locations to address fetching and finding tasks, and edges can depend on the actual embodiment of the robot in

Topological layer

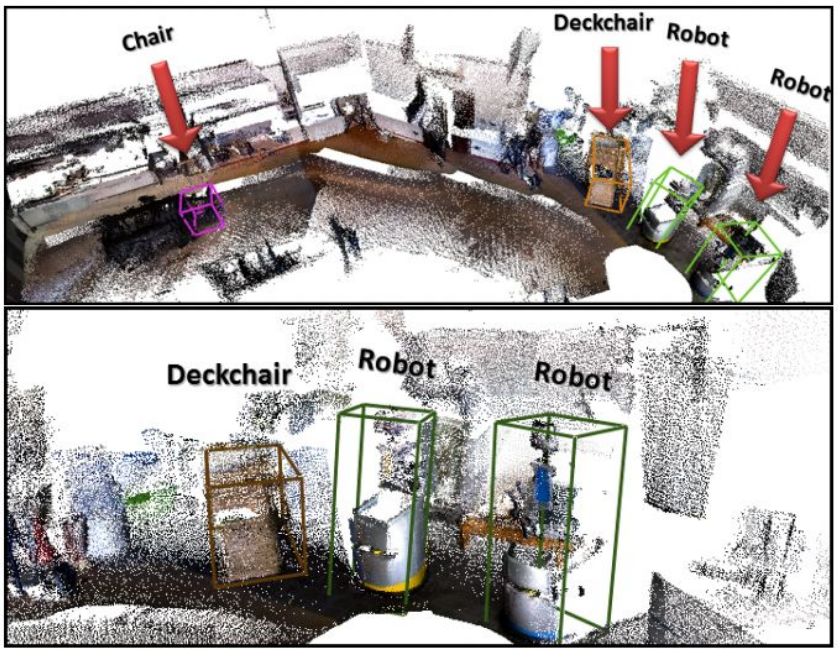


Figure 2.5. Semantic map of and indoor office. Objects have been labeled and anchored in the metric representation of the environment (image from [33])

order to encode the affordability of an action with respect to the platform used – traversability may vary depending the robot deployed.

Topological maps can be generated offline given a metric map as input [112, 38], or they can be generated by the robot during exploration [158]. This latter setting is however more appealing for an autonomous robot. In fact, even though it requires to simultaneously handle (1) incoming sensory information, (2) determine robot movements and (3) managing the state of the current topological graph, it allows to have an adaptable representation of spatial knowledge that can be structured in order to support the current mission.

This layer provides a better model of the structure of the environment but still, it remains related to the topology of the space, which mainly supports robot motions and task-oriented navigation. Hence, to provide a "human-compatible" understanding of the space, robots need to ground high-level semantic concept to their sensory observations and annotate spatial knowledge with a more rich meaning.

Semantic layer

This is generally referred to as the problem of semantic mapping. The semantic layer is key to enable robots to operate in human populated environment. It provides the information needed to understand and interact with humans. Such a layer is usually used to semantically label spatial entities (e.g. objects) and to express their attributes and relations. Different approaches provide various solutions to the semantic mapping problem. For instance, many techniques exploit ontologies to represent high-level concepts [168, 213]. Generally, these approaches are robust and reliable, but it is difficult to use them when there is not a complete knowledge about the environment before robot deployment. Differently, other approaches automatically discover such semantic properties at running time, either by relying

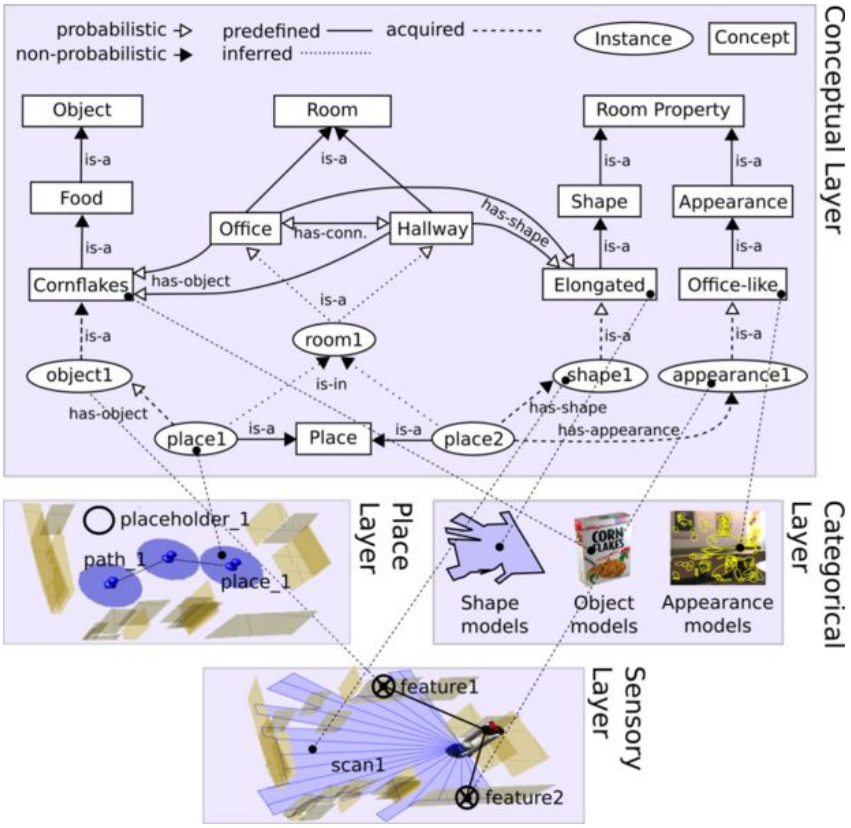


Figure 2.6. A hierarchical representation of spatial knowledge in an indoor environment. The representation spans different layers of abstractions and highlights relations among objects and places (image from [155]).

upon on-board sensors [18] or by assuming the so-called "man-in-the-loop" [70]. Nevertheless, despite the methodology to acquire new knowledge, such a layer is generally build on top the metric and topological ones. Thus, even if it provides human-level understanding of spatial knowledge, the first issue to be tackled is to ground and anchor semantic concepts back to the lower layers. For example, Figure 2.5 illustrates the result of back-projecting semantic labels of objects to a point cloud collected with robot sensors.

2.2.1. SK4R Representation Goal

Our contribution to spatial knowledge representation is to push towards a more holistic model of the environment, that is able to abstract perceptions into high-level semantic information, and to ground knowledge directly into the physical world [15, 161, 79]. Generally, existing frameworks represent the environment on a hierarchical structure that exposes the level of abstraction where the focus is [49, 155]. The goal, thus, is to build a general purpose multi-layered representation of knowledge that is expected to be used during mission by the agent. In [155] (see Figure 2.6), for example, the authors propose a four layers architecture, whose layers can be exploited to perform navigation, recognition and reasoning about entities and their relations.

The hierarchy analyzes the raw sensory information on the lower layer which detects important landmarks and ensure safe navigation. Higher in the hierarchy, the *place* and *categorical* layers support reasoning about object instances and how to navigate within the environment. Then, the top is represented by the *conceptual* layer that associates object instances, attributes and relations to the environment. Such a layer represent objects relations probabilistically in order to deal with uncertainty and partial observability. This structure is designed to enable robots to interpret the environment as humans do, in order to enable similar behaviors. However, the aim of such a representation is to represent the environment as it is, rather than modeling spatial knowledge to explicitly support robot actions.

Our approach follows this holistic view of spatial knowledge, but rather than focusing on modeling every aspect of the environment, we want to achieve a more compact and practical representation that is specifically designed to support decision making for robots in complex environments. Hence, as described in Chapter 4, we define the SK4R representation to demonstrate how an explicit representation of robot actions can improve performance in accomplishing various tasks.

2.3. Reasoning and Decision Making

An autonomous robot has to interpret the stream of sensory data, update its world representation, and behave in accordance with its goal specification. To this end, many researchers focus on the problem of generating complex behaviors by relaying on the current belief of robot. Often, this is achieved by employing graphical models for behavior generation. Depending on the problem requirements and environment characteristics, different models can be used to guarantee system performance. For example, finite state machines [64] have been largely used to model behaviors in environment with low uncertainty and relatively small state-spaces. However, in robotics this is a hard condition to satisfy, and the approach is rather based upon probabilistic models of reasoning, and lately, with models learned directly from robot experience. In particular, in the following chapters of this thesis, we rely upon the theoretical background of a specific graphical model, namely Markov Decision Processes (MDPs).

2.3.1. Markov Decision Processes

MDPs are a widely used mathematical framework designed to support planning and decision making. They are used to model processes where each state of the environment is the result of the dynamics of the environment itself, and the effect of robot actions. Within such a framework, agents behaviors are represented as a graph whose nodes are states of the system, and edges denote the agents' actions. MDPs exploit an important property referred to as the *Markov assumption*. Such an assumption formalizes the next state of the system s' to depend only on the current state s and the action a performed by the agent – and not an history of actions and states. The model is completed by associating to each state and action both a reward signal $r(s, a)$ and a transition model $\mathcal{T}$. The former is used to quantify the "goodness" of performing an action a in s, while the latter determines the effects of

applying a in s. Formally, and MDP is a tuple of six elements

$$MDP = \langle S, A, \mathcal{R}, \mathcal{T}, \gamma \rangle, \tag{2.1}$$

where:

- S denotes the state space that includes each possible state of the systems s such as $s \in S$;
- A is the set of discrete actions that includes all the actions a that the robot can perform such as $a \in A$;
- $\mathcal{R}(\cdot) \in \mathbb{R}$ is the reward function which can be usually implemented in three configurations depending on the desired reward signal. Namely, the reward value r (or *return*) can be computed by considering only the current state $R(s)$, the current state and the action performed $\mathcal{R}(s, a)$ (most common use), and the complete transition $\mathcal{R}(s, a, s')$ with s' representing the next state;
- $\mathcal{T}$ is the transition model governing the dynamics of the agent environment. It is usually expressed as a function $\mathcal{T} : S \times A \times S \longrightarrow [0, 1]$ representing the probability of a given transition to happen;
- γ is the the discount factor taking values in $\gamma \in (0, 1]$.

Moreover, rewards are also *Markovian* as they do not depend on the history of states traversed by the agent, and the selection of an action is taken in accordance with a action policy π, that is represented as a function $\pi : S \longrightarrow A$ mapping states to actions. In this thesis, we rely on *stochastic* policies that, given a state s, they generate the probability distribution over actions in s such that $\pi(a \mid s) \in [0, 1], \quad \forall a \in A$. To characterize such a distribution, we let the agent explore its state-space and to generate a dataset of state-action pairs $\zeta = (s_t, a_t)_{t=0}^{T}$, where T is the timestep horizon of each episode. Then the cumulative reward collected for each episode it defined as $\mathcal{R}(\zeta) = \sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t)$ representing the sum of rewards that the agent collects by executing a sequence of actions. The discounted policy π is then computed as the function that, given a distribution of states, finds the sequence of actions that maximizes the *expected cumulative reward* $\mathbb{E}_{\zeta \sim \pi}[\mathcal{R}(\zeta)]$ over T. Accordingly, the state-value function is formalized to represent this problem as:

$$V^{\pi}(s_t) \;\equiv\; r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots \;\equiv\; \sum_{i=0}^{\infty} \gamma^i \cdot r_{i+1} \tag{2.2}$$

where r is the reward collected at each visited state and the γ is the discount factor as previously introduced. The discount factor weights the importance of future rewards when evaluating the the current state of the system s at time t. Intuitively, when $\gamma \approx 1$ then future rewards weights equally as the immediate reward while, in the case $\gamma \approx 0$, only the immediate reward significantly influences the value function V^{π} at t. The value is used to evaluate the "goodness" of states traversed by the agent when executing an action policy π. Eq. 2.3, in fact, explicitly expresses the value function to depend on both the current policy and the transition model of

the environment. It represents the state-value as the return of the state s and the expected value of the next state s' reached by following the policy π:

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

Such equation exposes the action policy and allows us to compute the optimal action policy as the one that greedily maximizes the state-value function such as

$$\pi^{*} = \arg\max_{a}\{R(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s')V^{\pi}(s')\} \tag{2.4}$$

2.3.2. Action-state Value Function

The goal of the agent is, therefore, to learn an optimal mapping from states to actions. This, however, is a very difficult problem because the environment does not provide state-action pairs $\langle s, a\rangle$ to the agent, but rather rewards it with a *return signal* of the form $r_t = \mathcal{R}(s_i, a_i)$. Hence, it is easier to learn a probability distribution in the joint state-action space and generate an optimal policy from it. To this end, the Q action-value function is defined so that its value depends on the cumulative reward obtained by applying action a in a state s and the discounted value of the next state, such as

$$Q(s,a) \equiv R(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s')V^{\pi}(s')$$

where s' is the state reached by executing a in s, according to the transition model $\mathcal{T}$. It is important to notice that the action-value faction has the same form of Eq. 2.4 which, by solving their Bellman optimality equations [207], allows us to express the value function in terms of the action function as

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

$$Q^{*}(s,a) = R(s,a) + \gamma \sum_{s'} \mathcal{T}(s,a,s')Q^{*}(s',a') \tag{2.6}$$

and find the optimal policy by means of the action-value function

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

Eq. 2.7 enables the agent to choose among actions and not states. Intuitively, this is crucially important since the agent cannot decide in which state to be – especially in stochastic settings – while it can always deliberate which action to perform. In fact, such an equation makes possible to learn a policy π even if there is no knowledge about the transition model, and only the reward signal is available.

Partially Observable Markov Decision Process

In countless applications, especially in robotics, it is not possible to fully observe the true state of the world and the agent needs to behave by considering *state-dependent observations*. As an extension of the MDP, *Partially Observable Markov Decision*

Processes POMDPs are used to model partially observable environments so defined when a complete knowledge of the state of the world is not available. Similarly to Eq. 2.1, POMDPs are defined as a 8 element tuple

$$\langle S, A, O, \mathcal{R}, \mathcal{T}, Z, \gamma \rangle, \tag{2.8}$$

where $S, A, \mathcal{R}, \mathcal{T}, \gamma$ are defined as in Eq. 2.1, while O is the set of observations, and Z is the set of observation dynamics that model the probability of observing $o \in O$ in a given state $s \in S$. It is worth remarking that under the assumptions of partial observability the agent cannot have a complete understanding of the state of the environment. Thus this framework needs to exploit the notion of *belief state* and *belief space*. The former is a probability distribution over a set of states $b(s)$, while the latter is the set of all possible distributions. On the one hand, such a formulation allows to take into account the error and uncertainty in perceiving the current state of the world – which is a mandatory in robotic settings. While on the other hand, the problem to solve is more difficult as the belief space is significantly larger. The agent updates its beliefs as

$$b'(s) = \eta Z(s, o) \sum_{s'} \mathcal{T}(s, a, s') b'(s) \tag{2.9}$$

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

Stochastic Multi-agent Games

Throughout this thesis we will present different environments and applications where more than one robot have to cooperate and coordinate to achieve a common goal. To this end, we recall the mathematical framework of the stochastic game as an extension of Markov Decision Processes to the multi-agent scenario. A stochastic game is a tuple

$$\langle n, \mathcal{S}, \mathcal{A}_{1:n}, \mathcal{R}_{1:n}, \mathcal{T}, \gamma \rangle \tag{2.10}$$

where n is the number of agents, $\mathcal{S}$ is the set of states of the environment, $\mathcal{A}_j$ represents the set of discrete actions of agent j, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the stochastic transition function that models the probabilities of transitioning from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ when an action is taken from the joint action space $\mathcal{A} : \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$, and $\mathcal{R}_j : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function of agent i. In this setting, decisions are represented through agent policies π_j, that define the behavior of each agent j by mapping states to actions. Given a stochastic game, the goal of each agent consists in finding a policy $\pi_j(s)$ that maximizes its future reward with a discount factor γ, along the collective cumulative reward of the team. The agents in fact, collectively attempt to learn a policy that allows to converge to the *Nash equilibrium* [139]. This is an important concept whose applicability span different research fields. The *Nash equilibrium* is named after John Nash who, in one of his economics study introduces it as the configuration of the agents strategies such that, each agent has not incentive to change its policy regardless the actions of the other agents in the environment.

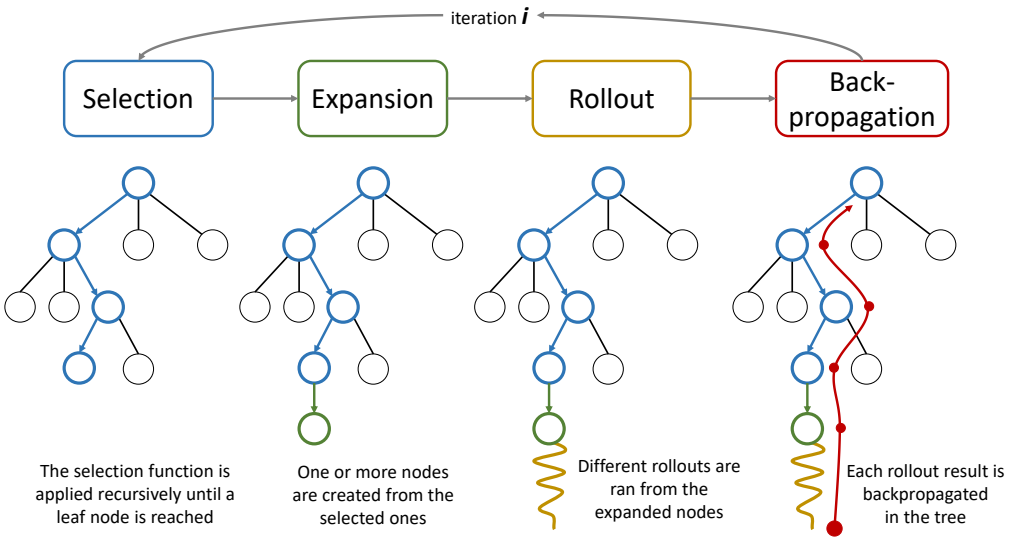


Figure 2.7. Four phases of the Monte-Carlo Tree search algorithm: (1) selection, (2) expansion, (3) simulation (rollout) and (4) backpropagation.

2.3.3. Monte-Carlo Tree Search Planning

Due to the complexity and the dimensionality of various applications, it is not always possible to address a decision problem by solving the optimality Bellman equation. For these reasons, several approaches that approximate state-action values with random sampling have been introduced. Among them, we report the *Monte-Carlo Tree Search* (MCTS) methods due to their recent successes in addressing decision theoretical planning in huge state-spaces [199].

MCTS methods work by iteratively construct a partial search tree whose estimates are improved after each visit. At each iteration, the algorithm goes through four phases that expand and evaluate each branch of the search tree. Figure 2.7 illustrates each of the algorithmic steps which are summarized as

1. **Selection.** In accordance with a *selection policy*, the agent decides which action to execute at each level of the tree, starting from the root until a leaf is reached.

2. **Expansion.** Each time a leaf node is reach, the algorithm has to check whether the leaf represents a terminal state. If it is the case, the algorithm does not expand that branch any deeper. Conversely, if the leaf is non-terminal then, it is expanded by evaluating all the actions that can be applied in that state, and by selecting one of them.

3. **Rollout.** During this phase, the algorithm follows a *rollout policy* that evolves the state of system until a termination condition is met. It is important to notice that such a policy can be arbitrarily chosen to balance exploration and exploitation.

4. **Backpropagation.** When a rollout policy reaches an ending state it is possible to evaluate the path followed by agent and the set of visited states along with.

This is done by collecting the outcome of the ending state, and propagating it back to the root by retracing the followed path. To guarantee statistical significance, the rollout and backpropagation steps can be executed a several times.

Iteratively, the MCTS algorithm builds a tree that is balanced toward nodes that *backpropagate* the best values up to the root. This allows the algorithm to explore portion of the state-space that are expected to lead towards more rewarding states and improve exploration [26]. Eventually, the algorithm performance depends upon how the nodes outcome are evaluated, and how they are propagated throughout the tree. In this thesis, we exploit a particular MCTS algorithm, the *Upper Confidence Bound for Trees* (UCT), that applies the UCB1 rule in the backpropagation phase. *UCT algorithm* In detail, the UCB1 rule computes – for each action applicable in a given state – the lower and the upper bound of expected reward. It is important to notice that, the UCB1 rule supports optimistic planning [28], since the algorithms only the actions with the best expected return to be expanded. Every time UCT finishes the backpropagation phase, it pairs an action to the current state which is used to update UCB1 values. UCB1 defines the confidence bound strategy [9] by balancing *UCB1 update rule* between exploration and exploitation. For each step $h \in [1, H]$ of the algorithm, and until the horizon is met, the algorithm selects the best action a^* to execute in the state s_h by means of the following equation

$$E = c \cdot \sqrt{\frac{\log \sum_a n(s_h, a)}{n(s_h, a)}} \tag{2.11}$$

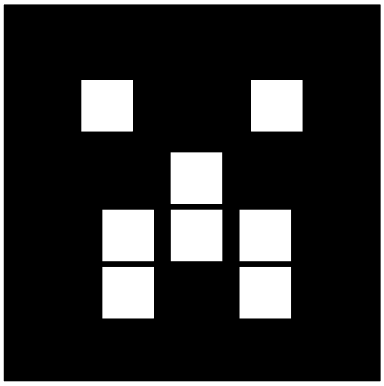
$$a_h^* = \arg\max_a Q(s_h, a) + E \tag{2.12}$$

where c is a constant controlling the exploration term, $n(s_h, a)$ is a function counting the number of times the action a has been selected in the state s_h and $Q(s_h, a)$ is the action value function. Intuitively, the exploration term promotes action that have been recently evaluated s_h while the Q value estimate favors actions that have backpropagated "good" estimates.

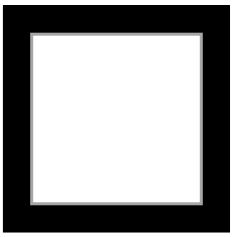
2.3.4. Deep learning

Recent advances in hardware technology have promoted and enabled a broad use neural networks, whose computational costs were earlier a prohibitive condition for any application. The improved scenario allowed for the development of a novel architecture able to capture and use a huge amount of information altogether, the so-called *Convolutional Neural Network* (CNN) or *Deep Neural Network* (DNN). DNNs represent a major breakthrough that significantly improved the overall performance of state-of-the-art techniques in the field computer vision and image processing [211, 114]. Lately, DNNs has been adopted in the field of reinforcement learning and decision making [134], by demonstrating a remarkable improvement in different applications.

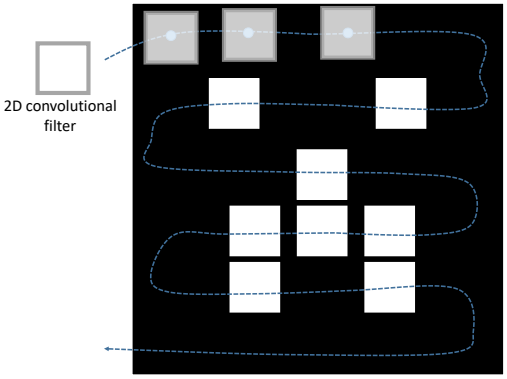
In this thesis, we exploit such a methodology and to generate action policies for different robotic applications. Thus, the following section recalls and summarizes its theoretical basis and introduces its formalization to address reinforcement learning settings.



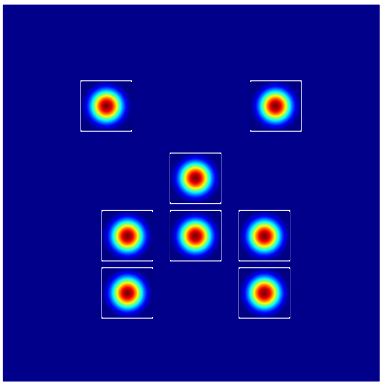
(a) Reference image



(b) 2D convolutional filter



(c) Convolution



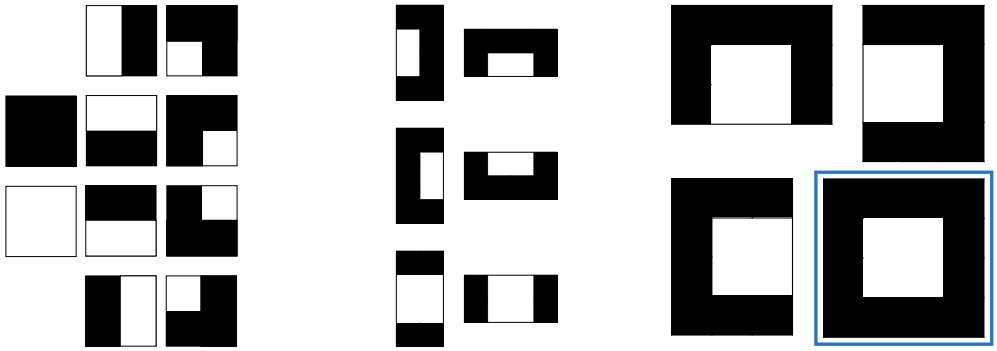
(d) Convolution output

Figure 2.8. White squares detector with a handcrafted 2D convolutional filter. The filter is convolved through the image to detect white squares in the image. The out put is a distribution over the image highlighting sections where the filter fired.

Elements of Deep Neural Networks

CNNs are able to extract information contained in an image and to elaborate features at different levels of abstraction. CNNs are represented as a hierarchical structure whose layers perform different mathematical operations on the input image. CNNs rely upon the concept of convolving a set of 2D filters over the image whose outputs are stacked and passed out to next layers of the network. To better understand how CNNs process images, let us consider the following example. Imagine that our goal is to spot white squares in a black image, Figure 2.8a. We can achieve such a goal, by manually tuning a convolutional 2D filter as in Figure 2.8b – where gray weights allow for uncertainty in the detection. Then, if we convolve the filter over the image (Figure 2.8c), we can spot white squares in the image (Figure 2.8d). What a CNN does is to learn those weights within the filters in a hierarchical manner. In particular, CNNs encode information with a more abstract meaning as in-depth the networks grows. For example, if a three layers network is used so solve the square example, it first leans how to detect simple descriptors at *layer 0*, such as edges, corners, plain colors (Figure 2.9a). Then it will use the output of the first layer to

Example



(a) first layer features **(b)** second layer features **(c)** third layer features

Figure 2.9. Features detected at different layers of the DNN. Lower layers detect more simple features of the image (e.g. colors, edges), while top layers can detect complex features with a significant semantic meaning (e.g. complete squares).

learn, at *layer 1*, specific spatial relations of descriptors extracted by the previous layer (e.g. relations among edges and corners, Figure 2.9b). In the last layer, it will receive these descriptors (corners and edge) and learn spatial relations of those. For example, as shown in Figure 2.9c, it might learn a filter that fires only when a set of four descriptors are paired and symmetrically displaced. This particular filter is highlighted with a blue square in the figure.

CNNs learn these features by analyzing a gigantic amount of data. They grow in depth to add more and more semantic meaning to features, and to be able to improve their generalization capabilities. Since CNNs can grow very rapidly, they are not solely composed of convoltutional layers, but also by *pooling* and *non-linear activation* layers. The former is used to reduce image dimensionality and limit the number of filters used, as long as the network grows in depth. The latter instead, is used to model non linearity of the dataset, usually implemented as a ReLu activation functions. Finally, each CNN terminates with a fully connected layer that performs the actual classification by receiving as input the output of the last "deep layer", which encodes the highest level of semantic knowledge extracted by the network. To learn the weights of the convolutional filters a *backpropagation* algorithm is generally used. The algorithm propagates the gradient error in classification from the output layer back to the input one, and adjusts weights of the network iteratively.

Structure

Learning

Deep Reinforcement Learning

Lately, the reinforcement learning community adopted CNNs to deal with prohibitively large state-spaces and solve many complex tasks. It this setting, the first success of deep reinforcement learning (DRL) is owned by DQN, a deep learning architecture introduced by Mnih et al. [134], in DeepMind research group [1]. The authors first challenged their approach on different Atari-2600 games (Figure 2.10 illustrates the structure of the network used), and then implemented a CNN to challenge the human expert of – what is known as – the most difficult game ever

[1] `https://deepmind.com/`

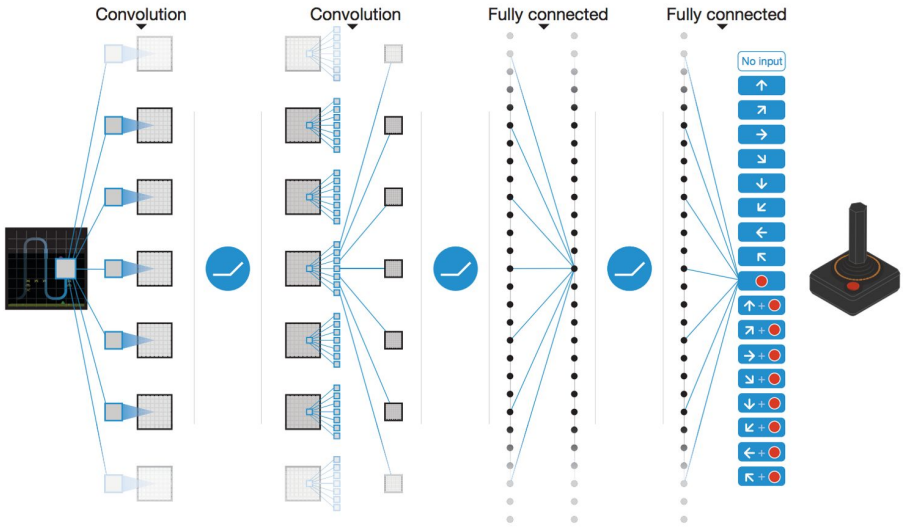


Figure 2.10. A visual representation of the DQN used to learn an action policy in an Atari-2600 game, image from [134].

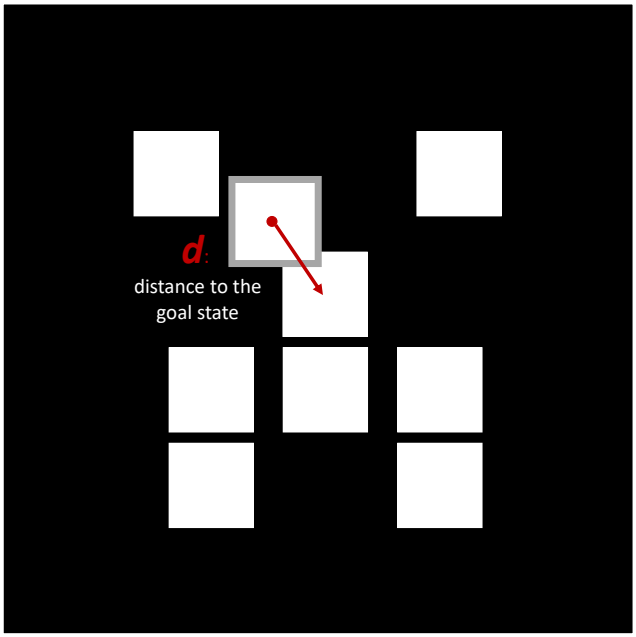


Figure 2.11. The goal is to overlap the filter with one of the squares in the image. The system can observe the current state (i.e., distance to the target square), and the reward signal with respect to the current state.

invented by humans, the game of GO. This was a huge achievement where the *deep* agent shown, for the first time, the ability to reason about next moves, and not just to select an action among a set of available ones. In this context, the input of CNNs as in the image classification scenario, but instead of classifying objects in an image,

their goal is to determine which action to perform in order to maximize a cumulative reward (Eq. 2.7). To better frame this shift in the objective, let us recall the previous example of finding the square in an image. This time the error is produced by the "distance" from the current state to the goal state in the form of a reward signal. For instance, if the task of the agent is to overlap its reference frame to the square (e.g. a tracking task), then we can express the distance to the goal state as an Euclidean distance in the image reference frame, see Figure 2.11. The goal of the agent is to select an action that maximizes the reward signal by reducing the current distance to the goal. At learning time, if an action is wrongly chosen then, the error computed with the reward signal is propagated throughout the network to update the weights and penalize that action for next learning episodes.

DRL represents one of the most promising techniques that can handle huge state-spaces and solve complex tasks. However, different research questions still need to be better understood, and approaches have to be improved in order to guarantee satisfactory performance in robotic settings. In this thesis, we address these problems and attempt to push the current research toward the realization of a framework that can exploit CNNs in a practical way and that enables robots to take advantage of it.

Chapter 3

Related Work

Manifold approaches contribute to the state-of-the-art to advance our understanding in knowledge representation and robot behavior generation. In this chapter, we analyze existing techniques by discussing their strength and limitations. Section 3.1 introduces existing frameworks to knowledge representation for autonomous robots. Then, Section 3.2 details approaches to behavior generation in the settings of both robot planning and learning. The section presents recent advances in deep reinforcement learning and introduces approaches in the field of multi-robot systems, which represent one of the major target application of this thesis. Finally, Section 3.3 relates our work to existing approaches, motivates our approach and highlights our contribution to the current state-of-the-art. It is important highlighting that this chapter serves to provide a global view of the research fields investigated through our dissertation. Then, for each of the research topics addressed in Part II, each chapter discusses state-of-the-art approaches more in detail.

3.1. Knowledge Representation

The goal of an effective "representation" is to play a set of *roles* (Section 1.1.2) to enable reasoning and action execution. We discuss method considering knowledge representation techniques and focus more on holistic frameworks attempting to provide the robot with general knowledge comprehending different spatial abstractions.

3.1.1. Spatial Representation in Robotics

We recognize the representation of spatial knowledge as one of the primary problem to tackle when developing a cognitive robot. In fact, these representation (1) inherently characterize the environment [70], (2) can be designed to encode task-related information [79], and (3) encode self-awareness [158]. Existing approaches detail the environment at different levels of spatial abstraction, by exposing important features (i.e. landmarks) that are key to aid the robot in different tasks. It is important remarking that different levels of abstraction extract features to support tasks requiring different cognition skills – spanning from localization to complex service tasks [38]. We categorize existing approaches to spatial representation in three major classes with different *spatial scope*: *metric*, *topological* and *semantic*.

Metric Maps

When designing an autonomous mobile robot, the first issue to consider is how to represent navigable space and obstacles. Intuitively, this problem has been addressed by proving accurate measurements and geometry of the space. First by engineering the representation of the space with respect to navigable areas – in a 2D setting – and then, by enhancing these representations to the 3D nature of the environment. This to allow the deployment of more complex robots such as aerial ones, and robot with the ability to manipulate objects. The problem of *metrically* mapping the environment and localizing in it, is known *Simultaneous Localization and Mapping* (SLAM). The SLAM problem is usually approached using a probabilistic formulation, due to the different sources of errors present in the sensors used during the map acquisition process – e.g. distortion errors in the camera sensor [50]. Over the years, different techniques have been proposed in the framework of Bayesian filtering, such as Extended Kalman Filters (EKFs), Extended Information Filters (EIFs) and Particle Filters leading to different kind of maps, from feature-based stochastic maps to dense maps like occupancy grid maps [216, 10]. To allow real-time execution of mapping algorithms, several methods rely on the early study of Lu and Milios [126] to represent the SLAM problem as a graph of spatial relationships between poses (i.e., nodes) of the complete robot trajectory [96, 117, 193].

Topological Maps

A topological spatial representation provides a more direct understanding on the structure of the environments, it highlights important areas and their connectivity. Specifically, topological maps provide a more coarse discretization of the continuous space that alleviates the computational and space demand of approaches employing metric maps. These maps allows for a more compact representation of spatial knowledge that can be more easily used to reason about navigation on global paths and local movements [215]. To generate topological maps, existing techniques rely upon a metric representation of the environment [122]. Usually, these approaches extract a Voronoi diagram encoding distance measurements from obstacles, both in the 2D and 3D settings [20]. However, as recent advances moved the focus on a more robot-centric point of view, the generation of topological maps is not only depending on the structure of the map itself, but also on robot perceptions. More recent approaches, in fact, focus on placing nodes of the graph in accordance with important landmarks determined from images of the robot camera [65, 222]. An important advantage of using topological maps is the possibility to bridge abstract concepts to nodes anchored in the metric world. This allows to support more complex behaviors and action planning, as well as reasoning about presence of objects, navigability, unexplored areas of the environment [158]. For example, Chung et al. [38] present a service robot capable of storing past observations into a pre-computed topological map in order to answer questions like *"Is there pizza in the kitchen?"* and *"Is the professor in his office?"*. Alternatively, in Zheng et al. [234] the authors learn the structure of the environment directly from its topology and robot observation. This allows the robot to generate a suitable representation built upon direct experience.

Semantic Maps

However, in order to build a functional robot systems able to operate and share environments designed for humans, we need to provide them with the ability to represent high-level spatial semantic concepts. This is known as the *semantic mapping* problem [113]. The semantic map serves to ground high-level semantic concepts into robot perceptions. To this end, existing frameworks span several spatial

abstractions [66] to formalize *semantic knowledge* and generate semantic maps [142]. In literature, two approaches are mainly employed: *fully automatic* and *user supported* map generation. The former solely relies on information extracted from robot sensors. The latter, also known as *human augmented semantic mapping*, exploits information from external users that actively supports the robot in acquiring knowledge about the environment. Even though the robot is able to autonomously recognize objects, the user help is exploited in grounding the corresponding symbols [15]. For example, the work by Zender et al. [232] proposes a system which is able to create conceptual representations of indoor environments. They consider a robotic platform which owns a built-in knowledge. In this case, the user role is to support the robot in place labeling. In [155], a multi-layered semantic mapping algorithm is presented. The algorithm combines information about the presence of objects and semantic properties related to the space, such as room size, shape and appearance. Whenever an user input is provided, it is combined as additional property about existing objects into the system. Differently, approaches that are explicitly supported by the user input have to feature multi-modal interaction scheme to conduct the interaction. In example, Kruijff et al. [115] present a system that aims at improving the mapping process by clarification dialogues between human and robot using natural language is introduced. Similarly, Gemignani et al. [70] introduce a system to generate semantic maps through multi-modal interactions. In this scenario, they use spoken languages to command the robot, and a vision system to enable the robot to perceive the objects that the user wants to identify, and label. They formalize the acquired environmental knowledge for enabling robots to ground high-level motion and navigation commands to a structured representation of a metric map enriched with user specifications [32].

Affordance-based Representation

Despite the particular *spatial* scope that the aforementioned representations have, their goal is to represent the environment in order to be compliant with how humans perceive space and act in it. However, for an autonomous robot these representation may result incomplete as they do not explicitly aid action planning. To "complete" these representations, we assume, and add, a different perspective to spatial knowledge. Our goal is to encode in a spatial representation what an agent can do with a spatial entity (e.g. open areas, objects), rather then the entity itself. In other words, its *affordance*.

Affordances have been originally defined as action *opportunities* that objects offer to agents [73] and, in robotics, they have been adopted to better represent objects and their related actions. This concept has also been recently extended to describe environments as a combination of spatial affordances to adapt robot behaviors. For example, in [59] and [127], the authors use spatial affordances to support robot movements in a navigation task and to improve the performance of a tracking system respectively. In Kapadia et al. [97] affordances are used for collision avoidance, while in Diego and Arras [53] they are used to navigate in crowded areas. However, these contributions only represent spatial affordances to improve robot *navigation* skills. Their approach cannot be generalized to action planning in a more broad sense nor implemented to support different tasks.

Moreover, the idea of improving robot policies based on object affordances [125,

100], or discovering object affordances given an initial policy [226] has been already investigated in recent works. For example, in [226] the authors exploit a simple policy to learn the affordance of an object to be pushed. In [125] and [100], instead, affordances have been respectively exploited to learn action primitives for the imitation of humans and for autonomous pile manipulation. Still, spatial knowledge representations in robotic either are focused on a detail representation of the environment, or, they are chained to aid the execution of specific target task. Motivated by this discussion, and by the need of a spatial knowledge representation to support *general* action planning, our goal is to design SK4R (Chapter 4) to overcome these limitations. We formalize SK4R to represent spatial concepts – traversing different levels of abstraction – along with spatial affordances in order to support general action planning. This is indeed our goal, we improve the formalization of *spatio-temporal affordances* proposed in Riccio et al. [171] in order to support general action planning [170], and policy improvement in various robotic applications [172, 177].

Akin to our use of affordance representation, *Semantic labeling techniques* [67] have the goal of annotating sensor images representing the environment. Usually, the output of these techniques is a segmented image, in which, each pixel has been assigned to a given semantic class [110, 42, 236]. For example, in the case of self-driven cars these classes can be *road*, *building*, *car*, etc. However, these methods are solely focused on representing spatial entities in the environment, and, they do not couple the representation of the external world with an explicit representation of the robot actions. Conversely, such a combined representation is the goal of SK4R and constitutes one of the main contributions of this thesis.

3.2. Robot Planning and Learning

A large body of work has been developed in the research field of decision making and behavior generation. The goal is to create an agent able to deliberate the right action in the right moment in accordance with its current belief of the world. Depending on the application and the task requirements, different techniques have been proposed to shape robot behaviors. In our setting, robot policy generation becomes a much harder problem to deal with, due to uncertainty, partial observability and dynamic environments [55]. Several works have been proposed that rely upon a probabilistic model of the robot acting in the environment. For example, Simmons and Koenig [201] address the problem of indoor corridor navigation by modeling an MDP in a partially observable setting, while in [202], the authors challenge a small team of robots, to perform a cooperative exploration by relying on spatial probability distributions mapped into the environment with the aid of a fine-granularity grid map. Alternatively, Raza et al. [165] provide a framework to model robot behaviors with *First Order Bayesian Networks* in order to deliberate robot decisions in a cooperative scenario. However, this class of approaches models specific dynamics and addresses particular robot applications with a "bounded" state-space dimensionality.

Monte-Carlo Tree Search

For these reasons, Monte-Carlo Tree Search methods [26] have been investigated to exploit model-based planning and learning in more complex scenarios [197, 199, 140]. In an exemplar application, Karaman et al. [98] introduce RRT* to improve the quality of the solution found by standard RRT algorithm in motion planning tasks.

In a different setting, Nguyen et al. [140] exploit MCTS to localize sources of sound and guide the robot throughout the environment. The work by Gelly and Silver [69] represents the first success in the challenging game of Go. These approaches have proven to be able to achieve remarkable results in modeling challenging tasks and perform effective planning [199]. However, MCTS has to visit each node of the partial search tree a significant number of times in order to iteratively generate action policies. Unfortunately, in robotics this is not a condition that can be always satisfied. Especially in large, complex scenarios characterized by a huge state-dimensionality, e.g. multi-robot scenarios. In fact, even though MCTS methods can be used to encode prior-knowledge [145, 61], they lack generalization capabilities as they are not able to relate similar nodes of the search tree.

Robot Learning

Reinforcement learning approaches show a much more effective generalization capability [95] in robotic applications [106, 14]. In this setting, unstructured environments and uncertain dynamics [224] are difficult to handle through handcrafted policies, that typically fail or must be refined [108]. Although designing effective policies is impractical in most of these scenarios, and learning techniques are typically demanding and time consuming [106] for problems with large state spaces. The computational demand can be alleviated by initializing a policy with expert demonstrations, that restrict the learning process to a promising hypothesis space [185]. For example, Kober and Peters [105] learn a ball-in-a-cup task by first initializing motion primitives through imitation, and then improving robot policies via episodic learning. Similarly, Argall et al. [8] take advantage of user tactile feedbacks to influence the learning process and refine a demonstrated policy. Konidaris et al. [109] initialize skill trees from human demonstrations, improving them over time through different learning episodes. Nikolaidis et al. [141] use model-based reinforcement learning and expert demonstrations to enable robots and humans to collaborate in a different object manipulation tasks. The robot shapes its policy in accordance with person characteristics in order to maximize their reward in the collaborative task. Kim and Pineau [102] propose an inverse reinforcement learning approach to learn *socially-acceptable* navigation skills of a robot in human crowds. The goal of their agent, is not to minimize the traveled distance, rather to increase humans' comfort when robot is reaching its destination. In a different scenario, Jun and Kenji [94] apply policy learning to solve a 2-DOF stand-up task for a robotic arm. They exploit Q-learning and actor-critic methods to learn both task decompositions and local trajectories that solve specific sub-tasks. However, also their procedure is not easily scalable to more complex scenarios. To apply these learning techniques in complex domains, a large dataset of *good-quality* expert demonstrations is generally required, that has to be efficiently mapped to the agent's action space. Unfortunately, this is not always possible due to the lack of (1) domain experts, (2) practical ways of providing demonstrations, and (3) action mappings from experts to agents (e.g. hyper-redundant robots).

Deep RL

Recent trends in reinforcement learning have shown improved generalization capabilities, by exploiting *deep reinforcement learning* (DRL) techniques. We recognize the first remarkable contribution of these techniques in the work of Mnih et al. [134]. The authors use a deep Q-network to learn directly from high-dimensional visual sensory inputs on Atari 2600 games. Similarly, Silver et al. [200] use deep value networks and policy networks to respectively evaluate board positions and select

moves to achieve "superhuman" performance in Go. In [135], instead, the authors execute multiple agents in parallel, on several instances of the same environment, to learn a variety of tasks using actor-critic with asynchronous gradient descent. Similar advancements have been shown in the robotics domain. For instance, Levine et al. [120] represent policies through deep convolutional neural networks, and train them using a partially observed guided policy search method on real-world manipulation tasks. Moreover, Rusu et al. [189] use deep learning in simulation, and propose progressive networks to bridge the reality gap and transfer learned policies from simulation to the real world. Currently, DRL methods represent the most promising technique in addressing complex applications with *sparse rewards* where state-of-the-art methods have been struggling in the past years. In this thesis, we exploit deep learning to make the robot able to generalize its behaviors to unseen and new situations. Unfortunately, planning and learning with deep networks is computationally demanding as it requires a huge number of heavy simulations. Hence, in order to apply such a technique to our scenarios, we contribute a novel approach to address such a limitation, and present a more practical approach in robotic settings.

3.2.1. Multi-Robot Cooperation and Coordination

When deploying a robot in general purpose applications (especially the ones considered here, e.g. domestic, industrial, health-care), we cannot assume that it will not communicate and interact with others – both humans and robots. Moreover, a carefully engineered cooperation among multiple operating agents has proven to improve both performance and success guarantees [233, 170]. For this reason, we consider the investigation of new techniques involving multiple agents to be crucial for future robot generations. Thus, we consider multi-robot scenarios one of the major target application of this thesis.

Such applications, however, constitute a complex scenario where multiple robots may exponentially increase the state space. In this thesis, we often refer to multi-robot domains as a complex environment, that represents a difficult challenge for existing techniques. Proposed approaches to Multi-Robot Systems (MRSs) have been analyzed in different survey papers. For instance, Dudek et al. [57] give a first taxonomy based on communication and computation aspects. Cao et al. [30] provide a categorization of multi-robot coordination frameworks. Following, Parker et al. [147] highlights the issues and research topics related to MRS systems, while Stone and Veloso [204] discuss the relation of MRSs and machine learning techniques. In [62] the authors provide a classification of multi-robot approaches focusing on coordination issues of a MRS, while in [170] the authors highlight differences in current MRS approaches from a new point of view. They categorize existing work in accordance with their assumptions on *Distributed World Modeling* and *Distributed Task Assignment*. We refer at *distributed world model reconstruction* (DWM), when the focus is on exchanging information that allows for building a global model of the world that integrates information that cannot be acquired locally by each robot (e.g. reconstructing a map of the environment). While, we refer at *distributed task allocation* (DTA), when the application is focusing on generating and optimizing the coordination criteria governing the team of robots.

World modeling

Approaches to distributed world modeling, typically rely on a metric representation of the surrounding scenario. For example, Zhou and Roumeliotis [235] match relative reconstructed maps with an EKF-based SLAM approach. Howard [90] employs particle filters to merge several maps carried out by each unit in the team. Roumeliotis and Bekey [187] consider a multi-robot localization scenario, where agents update their world state through mutual detection. Differently, Pereira et al. [152] exploit a team of robots to perform efficient exploration by representing the environment as a topological map. Similarly, in [80] the authors accomplish a cooperative coverage task by using a stochastic multi-objective optimization algorithm in an industrial setting.

Task assignment

The problem of distributed task allocation is expressed as the problem of relating a set of tasks to a set of robots. Many approaches in the literature consider the world model as "given" and suitably represented to run and evaluate the coordination algorithm. For instance, both in Okamoto et al. [144] and Corrêa [43], the authors use distributed constraint optimization methods to coordinate a team of robots in simulated grid world. Different approaches make use of decentralized POMDPs to accomplish cooperative navigation in simple grid worlds [4, 77], the *two agent-tiger* and a *box pushing* problems [17], and *prey vs. predator* games [162]. Similarly, Capitan et al. [31] formalize decentralized POMDP based on auctions in order to perform cooperative surveillance. Furthermore, *Market-based* techniques are well established approaches to optimize coordination algorithms, even without an explicit formalization of the surrounding world. Dias et al. [51], in fact, enable a team of robots to bid for a given set of tasks in a fully distributed way without any representation of the environment of the robots. The team of robots self-organizes in sub-groups and bids for resources. More recently, Luo et al. [128] introduce an iterative greedy auction algorithm to allocate task among the team. MacAlpine et al. [129] adopt market-based utility estimation to coordinate a team of robots in a cooperative-adversarial scenario. In a similar configuration, [194] adopt a path planning algorithm to preserve team formation among different teams of robots. The goal of the teams is to go through each other by maintain their respective formation and minimizing the risk of being damaged.

Multi-robot scenarios represent a complex environment where to challenge robotic systems. To this extent, we propose a complete coordination system [170], that distributively reconstructs the DWM, and adapts the coordination strategy to the current objective of the team with a DTA approach.

3.3. Contributions to the state-of-the-art

Generating effective robot policies in end-to-end systems is a challenging problem that requires knowledge and expertise in various research fields. In this work, we aim at presenting a spatial knowledge representation for robots, SK4R, that is designed to adhere different important features required to implement an efficient robotic system.

Contribution to spatial knowledge representations

SK4R provides a semantic abstraction of spatial entities to support decision making. Many of the existing frameworks do not explicitly support action planning and focus on a detailed representation of the environment – decoupled from the

robotic agent. We recognize the importance of representing spatial entities at different levels of abstractions [70, 155], and we build on top of existing frameworks to add a different perspective which exposes *spatial affordances* of the environment. Our contribution introduces a more "functional" point of view in representing the environment. Our work mostly relates to affordance-based representation of the space [97], however, we enhance proposed architecture by introducing a spatial representation intended to aid general task formalization and execution.

Contribution to multi-robot systems

In multi-robot system (MRS), approaches can be grouped in two classes with different scopes: approaches to improve accuracy *world modeling & reconstruction* and efficiency in *agents behavior coordination*. Within the scope of the former, MRSs are used to improve efficiency in collecting data and modeling the environment where the robots are operating [187]. This class of MRS applications often is deployed either with an implicit coordination protocols, or robots perform independent tasks by neglecting teammates. Within the latter class of MRSs, research contributes in developing coordination algorithm and optimize the collective coexistence of the whole system [144]. Conversely, these methods usually assume the world model to be static and/or fully known to the agents. In this context, we combine a dynamic distributed world model with a distributed market-based techniques for role assignment to improve the overall performance of the team. We also rely on a distributed world model, but we adopt a more abstract representation of the environment which is adapted according to the robotic application, thus offering a much lighter and generalizable level of environmental representation. We exploit such a representation to adapt the coordination system to the current situation and select the best team strategy. We contribute state-of-the-art approaches by proposing a flexible framework for robot coordination based upon dynamic and distributed world modeling. We formalize contextual knowledge through SK4R, in order to enable robots to recognize configurations of the environment as *context* and respond to them adaptively.

Contribution to robot planning

In addressing the problem of policy generation in robotics, *generalization* is typically addressed with function approximators [8] (e.g. gaussian mixture models, non-linear approximators). However, they do not allow for the use of prior knowledge, which can be inefficient and lead to dangerous situations. To overcome this issue, decision theoretical planning techniques, such as Monte-Carlo tree search, have been used to embed prior knowledge in learning problems. Nevertheless, they show difficulties in relating similar states [69] (i.e. nodes of the search tree). To this extent, we contribute by proposing a novel methodology that attacks the generalization problem in policy generation by enhancing the Upper Confidence Tree (UCT) algorithm with an external action-value function approximator that, is learned over-time and, selects admissible actions. While the UCT algorithm drives the robot in the exploration of the state-space, the external action-value function aids UCT in expanding only states with a significant return. After each UCT iteration, new training samples are used to update both the UCB1 values of the search tree and action-value estimates. We contribute by introducing such an algorithm which improves tractability of planning techniques in robotic settings.

Contribution to robot hierarchical planning

Task decomposition and hierarchical planning is key to enable efficient robot behaviors. Especially, in complex scenarios, where unpredictabilities and environment constraints prevent the generation of a satisfactory action policy due to a large state-space. To tackle this issue, we contribute a novel hierarchical optimistic planning approach to improve the exploration of the state-space at learning time, and alleviate the "curse-of-dimensionality". The proposed approach borrows the concept of Hierarchical Task Networks (HTNs) [60] to learn, and exploit, high-level representation of the robot actions, that are used to perform focused exploration during the *expansion phase* of the UCT algorithm (see Section 2.3.3 in Chapter 2 for major detail about the UCT algorithm).

Contribution to Deep RL

As recently proposed methods in the field of DRL show remarkable results, our goal is to favor the implementation of Deep RL techniques in robotic applications [135]. In this setting DRL techniques still have a limited application as they require considerable number of iterations and a big number of "good" training samples [200]. We propose a method to overcome this issue by means of a focused exploration that allows the agent to achieve competitive policies since first iterations of the algorithm. In particular, we combine DNNs with model-based planning techniques – which have shown the best results in field of robotics [61]. Similarly to existing methods, we capture the state of the environment as an image which is then used as input of the DNNs. However, we use those states to also construct a Monte-Carlo search tree that is used to generate model-based exploration of the state-space.

Part II

Spatial Knowledge for Robots

Chapter 4

SK4R: Spatial Knowledge for Robots

A fundamental cognitive skill that robots must have is the ability to interpret and represent the environment in order react to its stimuli and achieve their objective(s). However, this is a very complex problem that exploits the efforts of research communities both in the field of robotics and AI. The theoretical background of these research areas, as well as, the survey of related work in this thesis, give as an understanding of promising research directions – that have to be further investigated – and open our minds to new perspectives in representing the physical world of the robot. In fact, we assume a novel point of view in robot knowledge representation that enables effective planning by relying on a functional representation of the robot environment. Such a representation is designed to explicitly encode robot actions, support their execution and goal-oriented behaviors. To this end, we present SK4R by introducing its building concepts and key components (Section 4.1). Section 4.2 illustrates how SK4R can be implemented to learn the spatial representation of a *following* tasks and guide a robot in planning next moves. Finally, Section 4.3 analyzes and discusses the benefits of the SK4R representation.

4.1. Spatial Knowledge for Robots

Given a configuration of the environment, robots should have the ability to understand which actions can be performed and, most importantly, to which extent they can lead toward task completion. In other words, autonomous robots have to feature the ability to interpret the scenario also with respect to what it *offers*, rather than just what it *is*. Humans, subconsciously exploit this ability every time they plan for their actions. For example, a cup can be held differently depending on whether it is filled or not. Equivalently, in order to walk through a door, we place ourselves differently depending on (i) where the handle is, (ii) which side the door opens, (iii) if there is anybody traversing in the opposite direction. We shape our behavior in accordance with the state of the environment in order to understand which is the action that allows for task completion **and** that best supports the current configuration of the environment. This way of perceiving the external world gives the possibility to interpret the environment with respect to its functionality

rather that just represent its features as static entities.

In literature, this concept has been introduced in a psychological study by Gibson [73] as the *affordance* that objects carry with and provide to agents. To quote Gibson's work

"The affordances of the environment are what it offers the animal, what it provides or furnishes, either for good or ill. The verb to afford is found in the dictionary, the noun affordance is not. I have made it up. I mean by it something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment – Gibson,'79."

In his study, Gibson proposes a new point of view in perceiving objects and the environments itself. His goal is to describe the environment through functionally meaningful features that detail the opportunities that it offers to an agent. This notion has been accordingly adopted in robotics to provide a new representation of objects and the actions that they enable [103, 111, 146]. However, when considering the affordances of an environment, methods proposed in literature cannot be directly applied. Differently from normal objects, the state of the environment is highly dynamic and contains the state of the robot and other dynamic entities, such as humans. This inevitably leads to a more complex problem that requires a more flexible representation to support general action execution. To tackle this problem, we extend the concept of object affordance, to the case in which the entity to characterize is the environment itself. The goal is to provide the robot with an efficient representation that connects actions – and their affordances – directly to the operational environment. To this end, we exploit *spatial knowledge* (SK) to which we refer to as a functional representation of the environment of the robot. SK, in fact, is used to (i) provide an understanding about important aspects of the environment and (ii) the definition of particular configurations of it that support action execution and planning.

In the literature, several approaches address separately the formalization of accurate environmental representations [70, 115] and the generation of robot behaviors [59, 226, 55]. Conversely, by adopting Gibson's view [73], we assume a novel point of view in spatial knowledge representation. In fact, we enable the robot to posses a functional representation of the environment by means of affordances, which is explicitly intended to support effective planning. The spatial knowledge representation that we propose is designed to explicitly encode robot actions and support both their local execution and goal-oriented behaviors. To this end, we present SK4R, a spatial knowledge representation for robots operating in complex scenarios. Precisely, the goal of SK4R is two-fold: (1) it has to provide a representation of meaningful landmarks of the operational environment and (2) explicitly represent spatial affordances of robot actions in order to support action planning. Such features are key to the goal of this thesis, and to define SK4R as a representation of *spatial knowledge* that can be learned, updated and used by an autonomous agent to modify its own behavior.

In order for SK4R to feature these capabilities, we design it to collect (as input) observations from the environment E – or a representation of them – and generate *affordance semantics* (AS) of robot actions. Precisely, we define the *affordance semantics* of an action a as a representation of the environment, that highlights areas

of E affording the execution of a. In general, a representation of an environment inherently imposes constraints on important features of E. The domain in which these features take values is referred to the state-space S of E. Then, we define more formally *affordance semantics* of a as portions of the state-space $\bar{S}$ that afford the execution of a in E. Where $\bar{S}$ has to necessarily belong to the complete state-space S of the robot application as $\bar{S} \subseteq S$. Throughout this thesis we refer as to AS as introduced by Definition 4.1:

Definition 4.1. ***Affordance Semantics (AS)*** *of an action a is defined as areas of the environment E that support the execution of a. Given a representation of the state of the environment $s \in S$ and S its state-space, AS defines particular values of s, $\bar{S} \subseteq S$, that enable the execution of a and allow for the completion of a task τ.*

ASs are designed to annotate the environment and ground spatial knowledge directly into the workspace of the robot. To this end, we need to define a mapping that, receiving the current state of environment s_t at time t, can generate the ASs of robot actions – or a composition of them. Hence, we define the *affordance function* κ, as a function that maps observations about the state of the environment s_t to ASs of robot actions. However, in order to let the robot learn, or simply influence, how affordance functions shape AS in E, we need to expose structural parameters of these κ-functions that can be used to directly influence their output. We refer to these fundamental parameters as the κ-function parameters $\boldsymbol{\theta} \in \Theta$ taking value in the parameters space Θ, which depends of the specific implementation of κ. These parameters are considered as an input of the κ-functions as their configuration actively influences the affordance semantics of a robot actions, and their study is an important contribution of this thesis. We define affordance functions as follows

Definition 4.2. *κ is a function*

$$\kappa_{a,\tau} : S \times \Theta \rightarrow \mathrm{AS}_{a,\tau}. \tag{4.1}$$

$\kappa_{a,\tau}$ depends on the environment E, an action a and a task τ_t to be performed at time t. It takes as input a current observation of the state of the environment $s_t \in S$, and a set of structural parameters $\boldsymbol{\theta} \in \Theta$ characterizing the function. It outputs a representation of affordance semantics $AS_{a,\tau}$ that evaluates the likelihood of each area of E to afford a to complete τ_t in s_t.

The function $\kappa_{a,\tau}$ characterizes affordance semantics by evaluating areas of E where the tasks τ can be afforded. It generates, at each time t, a spatial distribution of AS that supports the completion of τ. Throughout this thesis, we often write κ to intend the general mapping from observation to AS for a given action a and task τ, κ_τ to express the AS of the complete task and κ_a to intend the output of a function $\kappa_{a,\tau}$ related to a particular action a

Figure 4.1 illustrates the SK4R representation. In blue, it highlights the set of parameters $\boldsymbol{\theta}$ while in red, the state of the environment $s \in S$. In green, the illustration of the κ-functions. Each κ_a takes as input the state of the environment and its set of parameters in order to generate an AS_a for a particular action a. The representation composes a series of κ_a to highlight the individual contribution of an action to the task completion. Accordingly, the κ_τ function defining the AS of the

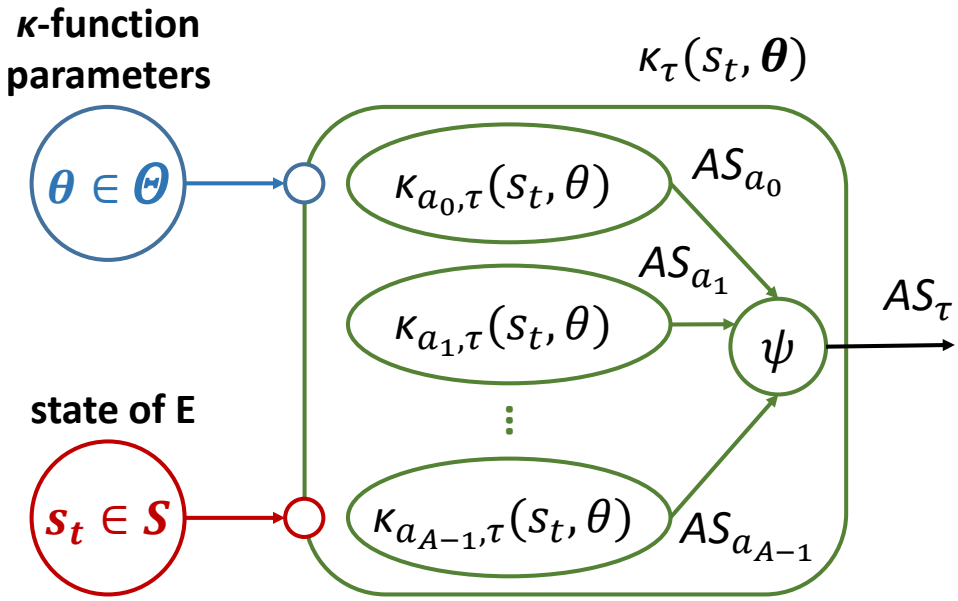


Figure 4.1. A representation for spatial knowledge for robots – SK4R.

complete task τ, is generated by considering the output of the different κ-functions unified by means of a function Ψ. The Ψ function, accordingly, assumes as input a set of AS_a, one for each κ_a function, and generates the AS_τ of the complete task that is expected to satisfy all its constraints and support its execution. The Ψ function can be chosen according to the specific of the application, nevertheless it has to ensure a mapping from a set of $\{AS_a\}_{a=0}^{A}$ to the affordance semantics in E of the whole task τ (where A defines the set of actions considered for a robotic platform).

It is important to highlight that we explicitly assume SK4R inputs to be (sub)modules of the representation as they are subjects of study of this thesis. Specifically, we refer to the module providing the current state of the environment as the SK4R environmental module (SK4R$_E$), and to the module maintaining the parameters of the affordance functions as the κ-function parameters module (SK4R$_P$). Their implementation – and adaptation to different scenarios – is formalized and detailed in following chapters of this dissertation. The study of these modules is fundamental to our aims. In fact, SK4R$_E$ is key to investigate how to structure the state of the environment and generate effective domain specific representations. SK4R$_P$, instead, represents the most direct way of influencing the affordance semantics. By explicitly connecting the generation of AS to structural parameters of κ-functions (i.e. $\boldsymbol{\theta}$), the agent can use them to learn and update affordance semantics, and to support the execution of its actions.

An important feature of a κ-agent is the ability to recognize particular states of the environment, and reuse acquired knowledge to address new, similar and more complex tasks – in the literature this is known as the ability of the agent to use "reflection" [22, 131]. We refer as to a κ-agent as an agent (e.g. a robot) that adopts the SK4R representation.

The modular representation of affordance semantics of SK4R inherently provides the possibility to do reflection. In fact, each individual κ_a function can be hierar-

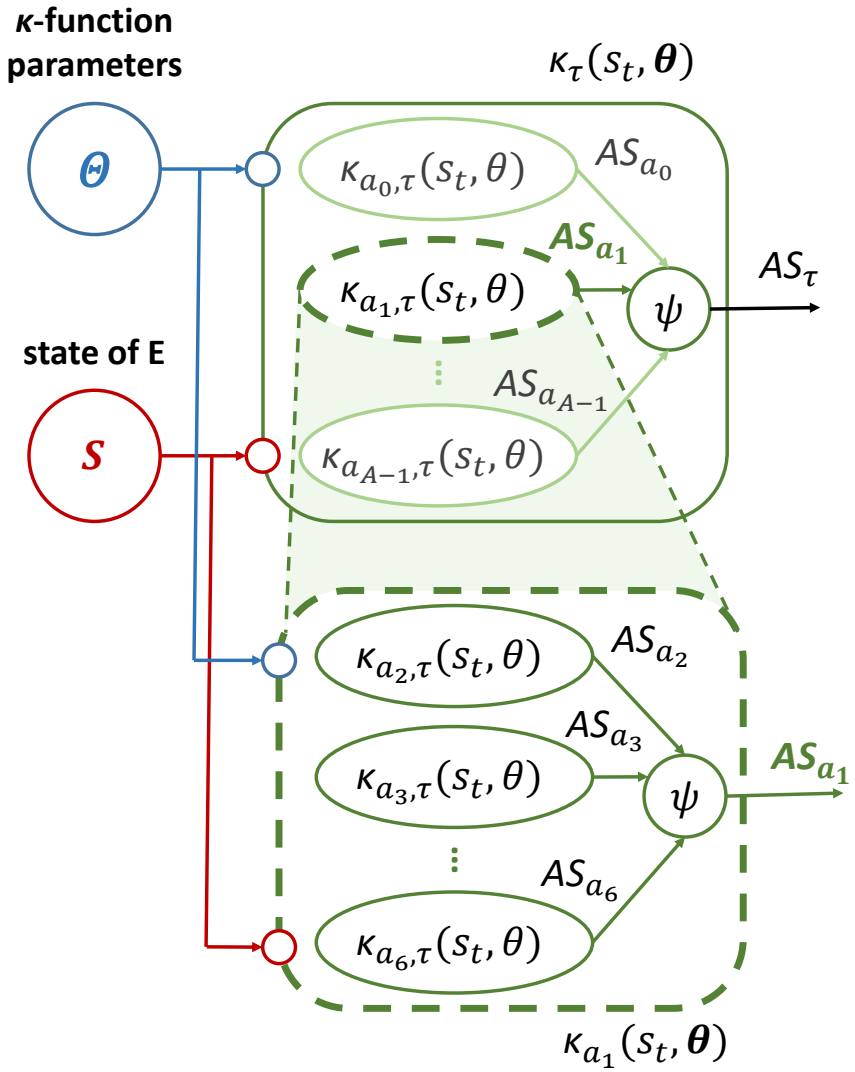


Figure 4.2. A representation for spatial knowledge for robots – SK4R. The representation is expanded to highlight its possibility to address complex task composition and reuse *action semantics* in different tasks.

chically represented by means of other κ-functions. Figure 4.2 frames this concept by visualizing the hierarchical representation of SK4R. Note that for each layer of SK4R the scheme is preserved, being the state representation and the parameters of the κ-function the inputs of the representation. It is worth remarking that this structure resembles the representation of Hierarchical Task Networks (HTNs) [60], where a task is arranged over a hierarchical structure, where, each layer represents the task at a different level of abstraction. From a more general representation of the task (top layers) to primitive actions (bottom layers). In HTN, at each layer, each action can be represented either as an high-level action (a set of primitive actions and other high-level actions) or a primitive action which enables to address re-usability and reflection. This property is key to our objectives and allows a κ-agent to rearrange and reuse κ-functions to generate composite tasks and their

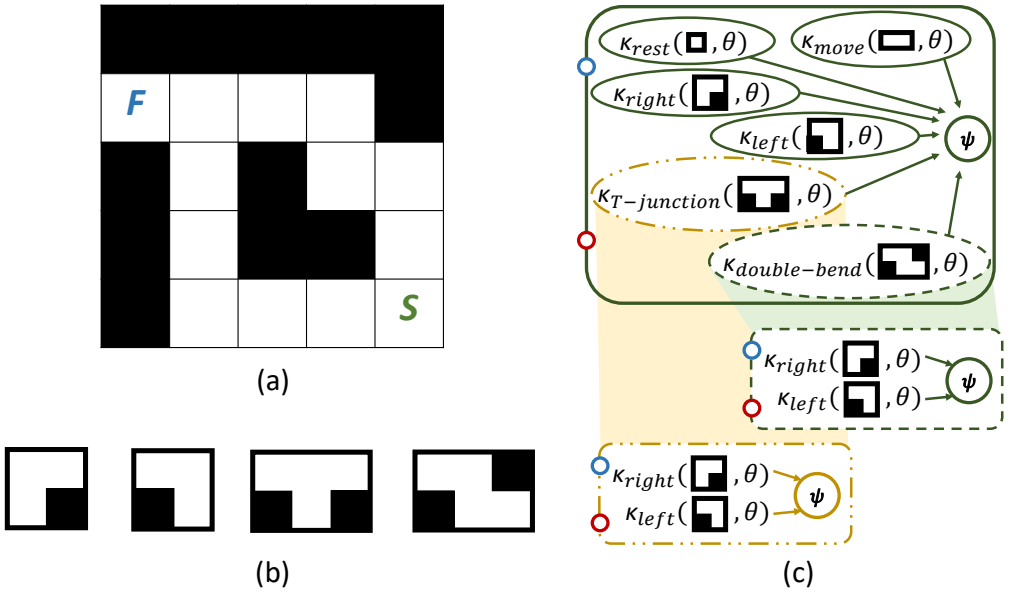


Figure 4.3. SK4R representation for a maze task.

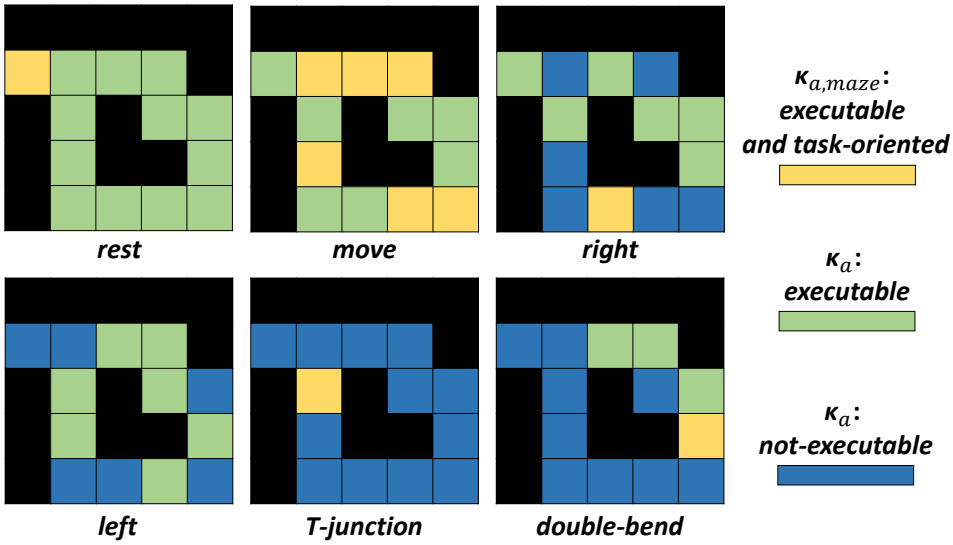


Figure 4.4. Spatial affordance semantics generate by SK4R to address the maze task. Each (sub-)figure highlights the output of each individual κ-function. In order from top to bottom and left to right, actions are `rest`, `move` (forward and backward), `left`, `right`, `traverse-T-junction`, `traverse-dobule-bends`.

affordance semantics.

To better understand how learned AS can be rearranged through the hierarchy, let us imagine a κ-agent that has to navigate a maze from *Start* (S) to *Finish* (F), as illustrated in Figure 4.3a. Let us also assume that the set of agent primitives is composed by motor torques that allow to turn left and right, move forwards and backwards, and stay still. As shown in Figure 4.3b, the environment features left and right corners as well as T-junctions and double-bends. Note that each

state in Figure 4.3b represents all the states in the scenario where a particular action can be executed. For example, the state s illustrating a right corner (left most state in figure) is pointing toward est, but it is used to also represent right corners pointing to north, south, and west. Equivalently, the other states represent all the configurations of E where their pattern can be recognized. This example allows us to highlight how κ-functions can be composed to address more complex actions. For example, both T-junctions and double-bends configurations can be addressed by composing the left and right actions. Accordingly, Figure 4.3c shows how primitives can be composed through the SK4R representation to generate composite actions and their ASs. The lower layer contains only the `left` and `right` actions that are necessary to compose both the `traverse-T-junction` (green) and `traverse-dobule-bends` (yellow). The top layer, instead, features the complete set of primitive actions plus the two composite actions generated from the low level. Such a structure allows to generate more complex actions whose execution can result to be more effective given the current state of the environment, and thus being a preferable option. Figure 4.4, conversely, shows the output of the six κ-function designed for this task. The top row highlights the AS of the `rest`, `move-straight` (forward and backward) and `turn-right` actions, while the bottom row the AS of the `turn-left`, `traverse-T-junction` and`traverse-dobule-bends`. The figure uses different colors to indicate through the AS where a particular state s of the environment E (in this case a grid of the maze) supports the execution of an action a – *green* if it does *blue* otherwise. Moreover, the AS supporting the execution of the navigation task is colored in *yellow* to explicitly highlight portions of the state-space where the execution of the action a leads to task completion. Finally, the κ-agent can exploit AS_τ to navigate the environment and perform situated decisions in accordance to the current state of the task, e.g. whether it is facing a right turn or a double-bend.

4.1.1. Generating Affordance Semantics

SK4R generates AS a representation of the operational environment that evaluates E with respect to the current state of the world and encodes areas of E where a particular task can be afforded. For instance, in the previous example, where the environment is represented as a grid-map, AS_τ encodes in each cell the likelihood of a given area to afford an action Figure 4.4. Nevertheless, according to Definition 4.2, the generation of AS_τ directly depends on a set of parameters $\boldsymbol{\theta}$ that modify how the κ-functions model the space. Hence, $\boldsymbol{\theta}$ constitute the main vehicle to shape spatial affordance semantics, and need to be carefully "designed" or "learned". In the first case, accurate understanding of environment and the task to address is required to shape the robot behaviors. While this approach is more direct and guarantees immediate performances in more constrained scenarios, it requires knowledge of a domain expert and the possibility to pre-schedule events of the environment. Conversely in the latter case, parameters of the κ-functions can be learned directly from observation. This is a more challenging scenario for general purpose applications, but it guarantees that the generated affordance semantics is shaped over the robot actions and characterized by direct experience with the environment.

The definition of AS and the way in which SK4R generates and exploits them

connects to the concept of *semantic-driven action selection* (SDAS) [3]. However, the semantic content exploited in existing SDAS techniques is not explicitly represented, and it remains hidden in latent variables of the system [184]. Moreover, semantic priors are usually defined before robot deployment and, they are not updated (nor improved) once given to the system [229]. Instead, SK4R features $\boldsymbol{\theta}$ to overcome this issue and allows the robotic agent to improve semantic representations over-time and through direct interaction with the environment. Chapter 8 illustrates how affordance semantics representations (ASs) can be learned and improved during robot operation. Throughout the following chapters, moreover, we analyze several approaches that differently implement the SK4R modules as well as various methodologies to design and learn the parameters of $\boldsymbol{\theta}$ the κ-functions.

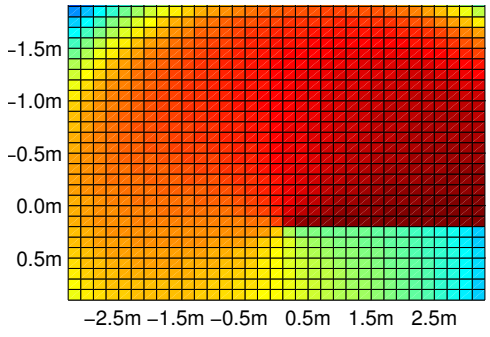
In order to illustrate the benefits of the SK4R$_E$ representation, next section details its implementation to address a following task (motivated in Chapter 1): an implementation of a generic κ_τ function that has to learn the AS of a following task.

4.2. Following Task

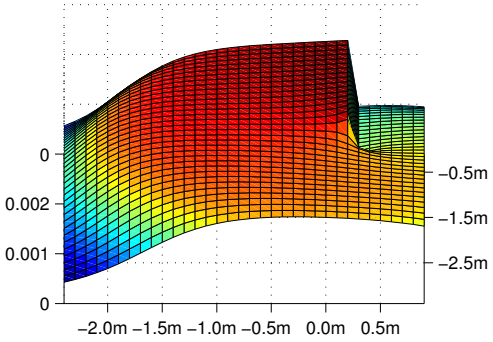
Despite the simplicity that it may suggest, a following task is a complex challenge to tackle where different environmental features have to be taken into consideration. In robotics such a task is used in different applications such as guidance [93], escorting [16] and support [83]. To correctly interpret such a task, the research community focuses on improving the detection of the target [12], or in generating recovering-behaviors in case of missing target [84], by preserving a safety distance when target is localized and tracked. In contrast, we propose to shape the robot behavior through AS_τ in accordance with the requirements of the task τ and the observation of the environment s_t at time t.

Hence, we perform an analysis of the learned AS to show the ability of SK4R to generate desired behaviors and influence them in accordance with s_t. In this exemplar case, we use expert demonstrations to teach a robot how to correctly interpret the environment. We evaluate the learned model by implementing the κ_{follow} function as a regression function, and by reporting the generated spatial action semantics of the following task AS_{follow}. Here, we consider two important aspects to characterize the environment, such as (i) the position of the person to follow and (ii) *risk areas*, i.e. areas near obstacles or behind a door. According to the representation of SK4R and Definition 4.2, we can generate AS_{follow} as the composition of two κ-functions. These functions first generate the AS of the two sub-tasks, namely keep a desired distance from the target and stay away from risk ares, and then, contribute to the generation of AS_{follow} by means of the Ψ function. In this scenario, we choose to implement Ψ as an intersection operator that includes in AS_{follow} only portions of E that simultaneously support the two sub-tasks.

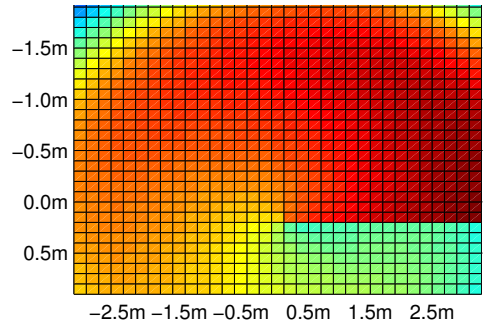
We encode in the state s_t the pose of the target to follow and the position of the nearest risk area. We use Gaussian Mixture Models (GMMs) and Gaussian Mixture Regression to represent and implement the function κ. The set of parameters $\boldsymbol{\theta}$ of the function is hence composed as a tuple $\boldsymbol{\theta} = \langle \pi_1, \mu_1, \Sigma_1, \ldots, \pi_N, \mu_N, \Sigma_N \rangle$, where π_i is the prior, μ_i the mean vector and Σ_i the covariance matrix of a mixture of N Gaussians.



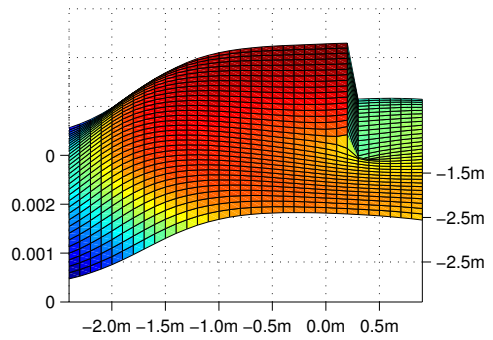
(a) first demonstration (top view).



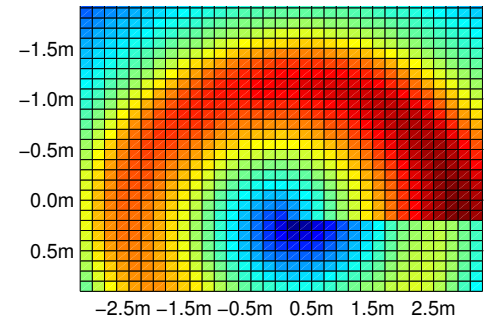
(b) first demonstration (side view).



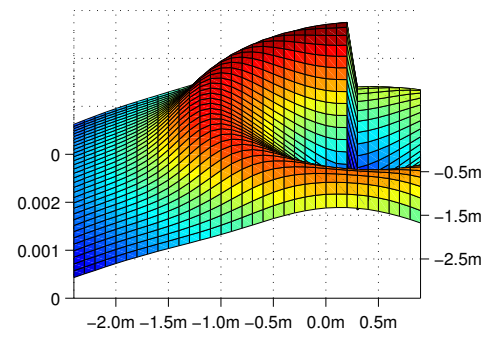
(c) second demonstration (top view).



(d) second demonstration (side view).



(e) third demonstration (top view).



(f) third demonstration (side view).

Figure 4.5. AS of a following task learned with increasing number of expert demonstrations. From top to bottom, the figures represent the side and top view of the model after the first, second and third demonstration. 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.

The $\boldsymbol{\theta}$ is learned from demonstration of different experts. To collect expert data we setup two robots in a simulated environment – one randomly navigates, the other is controlled by an expert through a joystick and follows the target robot by always moving between a minimum and maximum distance from it. During these sessions, the state s_t, as defined above, is recorded at each time instant

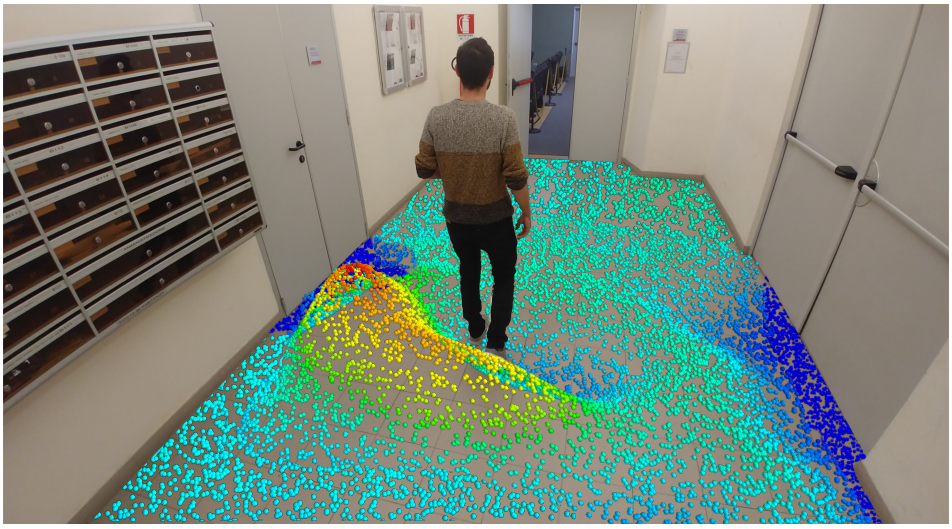


Figure 4.6. Spatial affordance semantics of two κ-functions to satisfy the task constraints.

together with the pose $\langle x_F, y_F, \alpha_F \rangle$ of the follower. The collected measurements are provided as input to the GMM and, by using Expectation Maximization, the tuple $\boldsymbol{\theta} = \langle \pi_1, \mu_1, \Sigma_1, \ldots, \pi_N, \mu_N, \Sigma_N \rangle$ that best fits the data is determined. In our experiments, prior to Expectation Maximization, the model has been initialized with k-means and a set of candidate GMMs has been computed with up to 8 components; the number of components has then been selected to minimize the Bayesian Information Criterion.

The learned model is used by the follower to determine, through Gaussian Mixture Regression, areas of E that enable the robot to execute the task and, hence, to generate AS_{follow}. In particular, the output of the regression consists of a mean vector and covariance matrix that enable us to infer the probability distribution (shown in Figure 4.5) of the follower pose, given the task τ. In this example, the κ-agent places itself to the desired distance while preserving a safe distance to risk areas, as required from task specifications. As shown in Figure 4.6, the AS_{follow} distribution explicitly captures the desired behavior that we want the robot to acquire. In fact, the distribution promotes areas that are behind the target to follow, and penalizes areas right behind a closed door which forbids the robot to navigate them. The learned model also encodes the preference of the expert demonstrations to follow a target from behind and on its left side (see Figure 4.5e-f). This is worth mentioning as it highlights also the possibility to embed prior knowledge into the design of the κ-function by relying upon social clues.

Finally, we report an analysis of the prediction error of the model generated by SK4R. To this end, we use expert data collected in three different demonstrations in an incremental fashion – after each demonstration we append new training examples to the previous dataset [185]. Then, we generate the model by splitting the dataset into two distinct parts. One is used to learn the κ-function, while the other is used to compute the error of the best pose, selected according to the learned model – against the expert behavior (i.e., the ground-truth). To evaluate our model, we ran the experiment 20 times. Figure 4.7 shows the mean and standard deviation of the

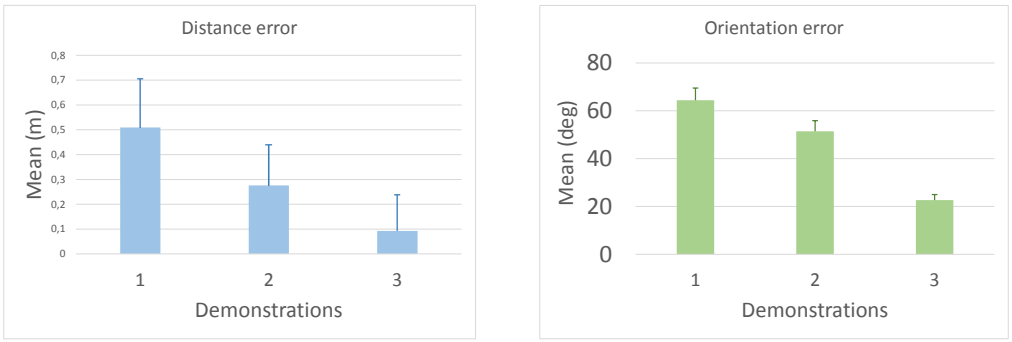


Figure 4.7. Error of the best pose, selected according to the learned model, against the expert behavior. On the left we report (a) the mean and standard deviation of relative distance error between the follower and the target, while on the right (b) the mean and standard deviation of the relative orientation error are shown. These values have been obtained by running 20 experiments and incrementally using three expert demonstrations (arranged on the x-axis).

prediction errors of the relative distance Figure 4.7a and orientation Figure 4.7b between the target and the follower position. It is worth remarking that, as soon as the model becomes more accurate (Figure 4.5), the prediction errors of both the distance and orientation decay.

4.3. Discussion

In the context of this thesis, we introduced *spatial knowledge* and defined *affordance semantics* (AS). We base our work upon these two concepts that we use to formalize SK4R, a novel representation for autonomous robots that are aware of their surroundings from a functional point of view. In fact, we design SK4R to preserve a canonical representations of the environment (e.g. metric, topological and semantic), but, we enhance such representations with knowledge about the functionality that characterizes the environment itself. This is a significant new perspective for an autonomous robot. In fact, by providing a connection between the environment and its functionality, spatial knowledge leads to a proper interpretation of it, and thus, to more efficiency in accomplishing a robotic task. To confirm the ability of SK4R to capture affordance semantics, we show-cased its implementation in a following task and validate the learned κ-function in terms of pose accuracy of the following robot. We set up a simulated environment where human experts could teach the robot how to correctly interpret the environment when performing a the task. After training, we let our system infer the best position to be with respect to the target and risk areas. Results show that (1) the mapping between the space and its AS is qualitatively valid and (2) the error generated by the use of our model decreases through the use of a larger number of expert demonstrations.

However, despite the promising results in the following task, different research aspects are left to be addressed during the description of SK4R. SK4R is intended

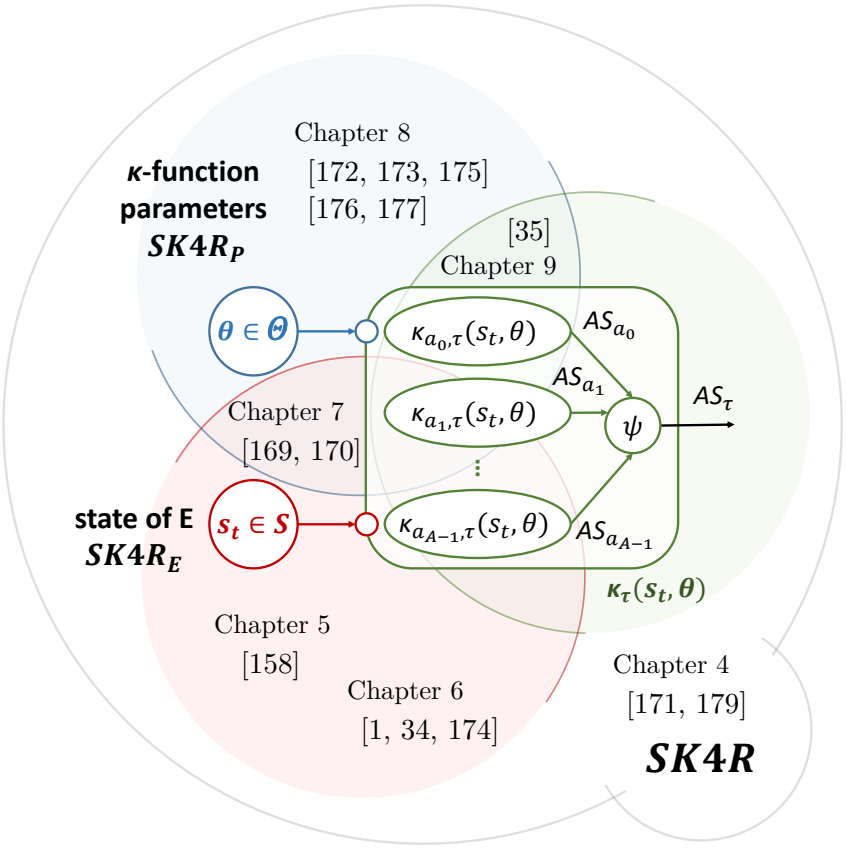


Figure 4.8. A representation for spatial knowledge for robots – SK4R. The working domain of each of the technical chapters is highlighted with different balloons and colors. References to chapters and related publications are added to provide a classification of the work of this thesis.

to be and end-to-end representation to support general action execution. Thus, its representation allows to traverse different research areas relating to robotics. In fact, as a result of the analysis and the formalization of our work, we focus our efforts on the realization of a working implementation of SK4R and its components. To this end, we face the realization of SK4R by highlight the benefits of a domain-specific representation capable of supporting action planning. Hence, we recognize the following set of research questions to be key to the implementation of SK4R:

1. Is it possible to define a domain-specific representation of the environment that allows the robot to encode affordance semantics at different levels of abstraction of spatial knowledge?

2. Can we use SK4R$_E$ to characterize social configurations of E? Robots operating in human-populated environments have to feature the ability to represent features not directly observable through robot sensors. Hence, it is often necessary to investigate how to formalize such social configurations.

3. The robot behavior is actively adapted to the state of E, however, is it

possible that the robot behavior influences how the state of the environment is generated? How can such a mutual dependency be implemented in a robotic system? Is it possible, for the robotic system, to feature an active interaction of the SK4R$_E$ and SK4R$_P$ components?

4. Given a representation of the state of the environment, can the robot improve its behavior over-time? In other words, is it possible to update (or learn) $\boldsymbol{\theta}$ parameters to improve the expressiveness of the AS representations?

5. We illustrated how SK4R can hierarchical define κ-functions to address reflection [131]. This lead us to investigate whether it is possible to learn hierarchies of κ-functions to support task decomposition by reusing ASs of primitive actions?

Each of the following chapters takes over one of these questions, and thus, it focuses on a particular research aspect related to the implementation of SK4R. Figure 4.8 associates each chapter – thus each of the previous questions – to its working domain within the SK4R representation. Additionally, the figure also highlights our individual contributions related to each chapter. Note that the interaction among SK4R$_E$ and the κ-functions excludes the SK4R$_P$ module. Thus, we consider that interaction to be out of the scope of this dissertation. In fact, without the $\boldsymbol{\theta}$ parameters the robot is not able to influence affordance semantics. Conversely, the study of the κ-functions can be interesting to understand the scalability of SK4R to different numbers of affordance functions and/or size of the state-space of the robot application. However, while we consider such a problem to be worth investigating, it is not key to a first formalization of SK4R. Hence, we remind its study to future work.

Nevertheless, despite the different research questions addressed by each chapter, our mission is to demonstrate the flexibility of SK4R in various and different applications. In fact, each of the following applications implement both the environmental module and the κ-function parameters module depending on the scenario requirements. Our goal is to demonstrate that an accurate connection between spatial knowledge and robot actions can improve the efficiency of a robotic system, and thus, push the current state-of-the-art towards more sophisticated skills for autonomous robots.

Chapter 5

SK4R for Indoor Robots

In this chapter, we investigate how to define a domain-specific state representations that allows to encode affordance semantics at different levels of spatial knowledge abstraction. An efficient state representation is a fundamental component for an autonomous robot to enable planning in unstructured human environments. In case of mobile robots, moreover, spatial knowledge that constitutes the core of the state, and directly affects the performance of the planning algorithm. In this chapter, we describe a spatial hierarchy to formalize the environmental module of SK4R (SK4R$_E$), which is used to provide the robot with the state of the environment $s_t \in S$. In particular, we propose, a probabilistic representation of spatial knowledge, spanning multiple levels of abstraction from geometry and appearance to semantics, and leveraging a deep model of generic spatial concepts. We design the implementation of SK4R$_E$ in order to represent space from the perspective of a mobile robot executing complex behaviors in the environment. In Section 5.1 we explain the principles behind this domain-specific implementation for indoor robots, and present its initial realization for a robot equipped with laser-range sensor (Section 5.2). Finally, Section 5.3 validates our implementation and demonstrates that SK4R$_E$ successfully builds a representation of large-scale environments, and leverages the deep model of generic spatial concepts to infer latent and missing information at all abstraction levels. [1]

5.1. Spatial Representation for Indoor Environments

Recent advancements in robotics have been driven by the ultimate goal of creating artificial agents able to perform service tasks in real environments in collaboration with humans [11, 79]. While significant progress has been made in the area of robot control, also thanks to the success of deep learning [120], we are still far from solving more complex scenarios that require forming plans spanning over large spatio-temporal horizons.

In such scenarios, domain-specific state representations play a crucial role in determining the capabilities of the agent and the tractability of the solution of

[1] The work contained in this chapter has been subject of study during my visiting period abroad at the University of Washington, Seattle (WA), USA.

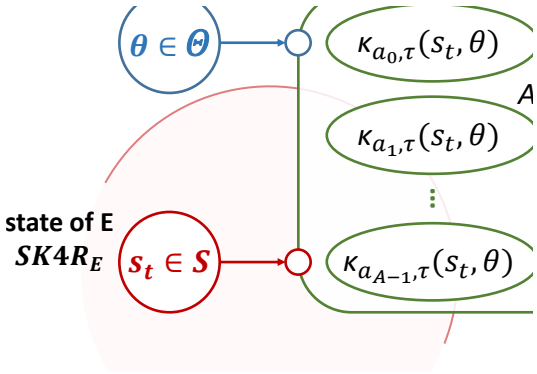


Figure 5.1. SK4R environmental module – SK4R$_E$.

planning algorithm. In fact, in case of mobile robots operating in large-scale environments, the way in which spatial knowledge is represented directly affects the actions the robot can plan for, the performance of the planning algorithm, and ultimately, the ability of the robot to successfully reach the goal. For complex tasks involving interaction with humans, the relevant spatial knowledge spans multiple levels of abstraction and spatial resolutions, including detailed geometry and appearance, global environment structure, and high-level semantic concepts. Representing such knowledge is a difficult task given uncertainty and partial observability governing real applications in human environments.

We propose a domain-specific implementation of SK4R$_E$ for an autonomous robot operating in an indoor scenario. Figure 5.1 recalls the overall SK4R scheme and highlights (in red) the subject of study of this chapter – SK4R$_E$. In detail, we propose a probabilistic representation of spatial knowledge designed to provide practical access to the world state $s_t \in S$ and to facilitate planning and execution of complex behaviors. Thus, SK4R$_E$ encodes the belief about the state of the world. However, more importantly, it also provides the feature to store and maintain information about spatial affordances. It does so by leveraging a hierarchy of sub-representations (layers) where, each layer, represents multiple spatial knowledge abstractions (from geometry and appearance to semantic concepts), using different spatial resolutions (from voxels to places), frames of reference (allo- or ego-centric), and spatial scopes (from local to global). The goal is to represent the environment – in accordance with Definition 4.1 – in a way that directly corresponds to how it will be utilized by the robot and its planning algorithms. The complete representation of SK4R$_E$ for indoor autonomous robots is illustrated in Figure 5.2. Specifically, SK4R$_E$ for indoor large environments is arrange on a four layers architecture. The *perceptual layer* is the lower layer of the representation and elaborates sensors readings. The *peripersonal layer* represents spatial landmark information and affordances in the space immediately surrounding the robot. The *topological layer* builds upon the previous two layers and is used to represent global topology of the environment and *navigation-related* affordance semantics. Finally, the *semantic layer* relates instance knowledge, processed through the lower layers, to human semantic concepts. Finally, the four layers communicate by means of the probabilistic *deep default knowledge model* (shaded purple columns), which provides definitions of generic spatial concepts

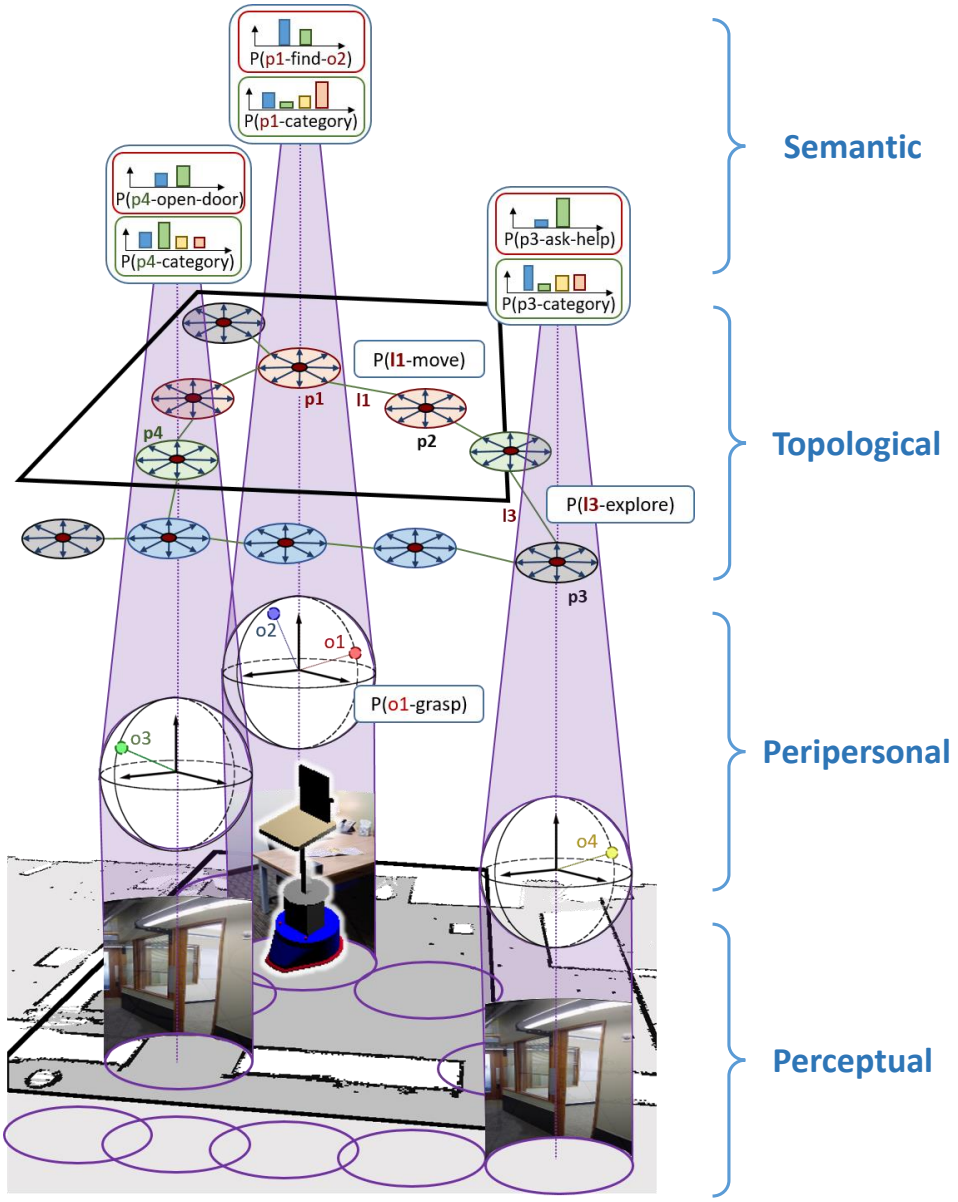


Figure 5.2. Multi-layered architecture of SK4R$_E$. The *perceptual layer* integrates perceptual information from the robot sensors. The *peripersonal layer* represents object and landmark information and affordances in the space immediately surrounding the robot. The *topological layer* encodes global topology and coarse geometry and navigation action affordances. Finally, the *semantic layer* relates the internal instance knowledge to human semantic concepts. The four layers are connected by the probabilistic *deep default knowledge model* (shaded purple columns), which provides definitions of generic spatial concepts and their relations across all levels of abstraction.

and their relations across all levels of abstraction.

SK4R$_E$ comprises both *instance knowledge* – knowledge about a specific environment (e.g. there are two tables in this kitchen) – as well as, *default knowledge* about generic human environments (e.g. the kitchen is used to cook food). The latter is

modeled using a recently proposed Deep Generative Spatial Model (DGSM) [160]. Specifically, DGSM leverages recent developments in deep learning, by providing fully probabilistic, generative model of spatial concepts learned directly from raw sensory data. DGSM connects the layers of our representation, enabling upwards and downwards inferences about spatial concepts defined at different levels of abstraction. For example, in the following sections, we show-case how the DGSM connects the different layers of SK4R$_E$ to infer the room type of a large indoor environments directly from sensor readings. Vice versa, we also show how such a connection is bi-directional and can be used to infer missing information from semantic labels of objects. In fact, such an implementation of SK4R$_E$ is designed to explicitly represent and fill gaps in spatial knowledge due to uncertainty, unknown concepts, missing observations or unexplored space. This brings the possibility of using the representation in open-world scenarios, involving active exploration and learning.

In this chapter, we first describe the general architecture of the SK4R$_E$ and present a realization of it for a mobile robot equipped with a laser range sensor. We then perform a series of experiments demonstrating the ability of the representation to perform different types of inferences, including bottom-up inferences about semantic spatial concepts and top-down inferences about geometry of the environment. We showcase its ability to build semantic representations of large-scale environments (e.g. floors of an office building).

5.1.1. Domain-specific Representation for Indoor Scenarios

Figure 5.2 illustrates the general overview of the SK4R$_E$ architecture. It represents the robot environment using four sub-representations (layers) focusing on different aspects of the world, encoding knowledge at different levels of abstraction and spatial resolutions as well as in different frames of reference of different spatial scope. The characteristics of the layers were chosen to simultaneously support both action planning and spatial understanding for the purpose of localization and human-robot interaction. In particular, the former objective is realized by directly encode spatial affordances.

SK4R$_E$ is organized as a hierarchy of spatial concepts, with higher-level layers providing a coarse, global representation comprised of more abstract symbols, and lower-level layers providing a more fine-grained representation of parts of the environment anchored to the higher-level entities. The layers are connected by a crucial component of the representation, the probabilistic *deep default knowledge model* (highlighted in purple in Figure 5.2), which provides definitions of generic spatial concepts and their relations across all levels of abstraction. The aim of SK4R$_E$ is to provide a practical representation of the world state that can be exploited to support robot operation. Therefore, it is worth highlighting that the hierarchy directly relates to a similar, hierarchical decomposition of the planning problem. A global planner can derive a navigation plan relying only on the top layers for representing its beliefs, a local planner can be used to plan specific manipulation actions using intermediate layers, with a controller realizing them base on knowledge in the lowest-level representation. Following a description of each component of SK4R$_E$:

- **Perceptual Layer.** The layer maintains an accurate representation of the

geometry and appearance of the local environment obtained by short-term spatio-temporal integration of perceptual information from sensors with finite horizon. Spatial information in perceptual layer is represented in an metric reference frame, which facilitates integration of perception from multiple viewpoints and sensors. However, the representation is always centered at the current location of the robot, and its range depends on the robot sensors (essentially a sliding window). Information outside the spatial scope is forgotten, which makes the layer akin to a working memory, and enables consistent large-scale higher-level representations without the need to maintain low-level global consistency. The layer provides a more complete input for further abstractions.

- **Peripersonal Layer.** Above the perceptual layer is the *peripersonal layer*, which captures spatial information related to object and landmark instances from the perspective of the robot. To support planning, the layer represents object affordances related to actions that can be performed by the robot. This includes manipulation (e.g. possibility of reaching/grasping an object or pressing a button), interaction in relation to objects (e.g. possibility of pointing to objects), and epistemic affordances (e.g. possibility of observing an object). Furthermore, the layer serves as an intermediate layer of the deep default knowledge model, used to generate descriptions of locations in terms of higher-level concepts (e.g. room categories or place affordances). While recent results from neuropsychology suggest existence of local, body-centered representations in animals and humans [88], our motivation for such a layer is to provide the robot with the ability to reason about its immediate surrounding and support local behaviors.

- **Topological Layer.** The topological layer provides an efficient representation of large-scale space, including coarse geometry and topology, and serves several key roles in SK4R$_E$. First, it provides a way to express the global pose of the robot. Second, it captures navigation and exploration action affordances associated with locations in the environment. Third, it is a global counterpart to the local peripersonal representations and anchors them in the large-scale space. Finally, it captures internal descriptors of places and serves as an intermediate layer of the deep default knowledge model used to derive semantic place descriptions. To this end, the layer performs a bottom-up discretization of continuous space into a set of locations called *places*. Places correspond to locations in the environment previously visited by the robot, and are meant to represent space at a resolution sufficient for action execution, while maintaining efficiency and robustness to dynamic changes. In other words, the resolution is selected to ensure that high-level navigation can be planned using the topological layer only, with local behaviors planned using the knowledge in the peripersonal layer at the destination. Places are spatially related to other, neighboring places, which encodes coarse global geometry of the environment and allows for path integration. For each place, the topological layer maintains a set of discrete headings, called *views*. Together with places, views can be used to represent a coarse pose of the robot – both topological positions and headings. Besides places and views, the layer also defines *paths* connecting

neighboring places into a topological graph. The semantics of a path between two places is the possibility of navigating directly from one place to the other (i.e. affordance semantics of navigation actions). Existence of a path in the graph, however, does not necessarily imply that it has previously been traveled by the robot. In fact, a path can indicate the possibility of navigating towards *unexplored space*. To this end, the topological layer utilizes the concept of "placeholders" [157], which can be seen as candidate places, and are used to explicitly represent unexplored space. As a result, paths that lead to placeholders express the possibility of epistemic exploration actions. This can be used to address open world problems [79] in the continual planning paradigm [143].

- **Semantic Layer.** On top of SK4R_E is the semantic layer, a probabilistic relational representation relating the spatial entities in the other layers to human semantic spatial concepts defined in the deep default knowledge model. This includes such concepts as object categories and attributes, place attributes, room categories, or the concept of a room itself. It is the semantic layer that captures the knowledge that an object is likely to be a cup, or that certain places are likely to be located in a kitchen. Furthermore, the layer plays an important role in planning complex tasks, by representing place affordances related to human interaction as well as actions characterized in terms of human concepts. For instance, it is the sensory layer that defines the affordance expressing the possibility of asking a person for help with making coffee or the possibility of finding a cup at a certain place. Finally, the layer enables transfer of knowledge from humans to the robot (e.g. capturing object category information provided by the user). Such knowledge can be utilized by the default knowledge model to generate lower-level information stored in other layers.

- **Deep Default Knowledge.** The four layers representing knowledge about the specific robot environment are linked by the *deep default knowledge model*. The model provides definitions of generic spatial concepts, valid for typical human environments, and their relations across all levels of abstraction (from sensory input to high-level concepts). This includes robot-internal models of objects in terms of low-level perception, places in terms of objects, place and object affordances, or models of semantic categories and attributes of objects and places. In other words, the four layers can be seen as defining the traditional ABox of our spatial knowledge base, while the deep default knowledge model represents its TBox. The role of the default knowledge model is to permit inferences about missing or latent aspects of the environment in each layer, based on the knowledge available in other layers. This includes bottom-up inferences (e.g. about semantic descriptions based on perception) and top-down inferences (e.g. about object presence or place affordances based on semantic descriptions). The resulting knowledge base constitutes a more complete (albeit uncertain) belief state for the planner. In this work, we implement this component using a deep generative probabilistic model based on Sum-Product Networks (see Section 5.2.5).

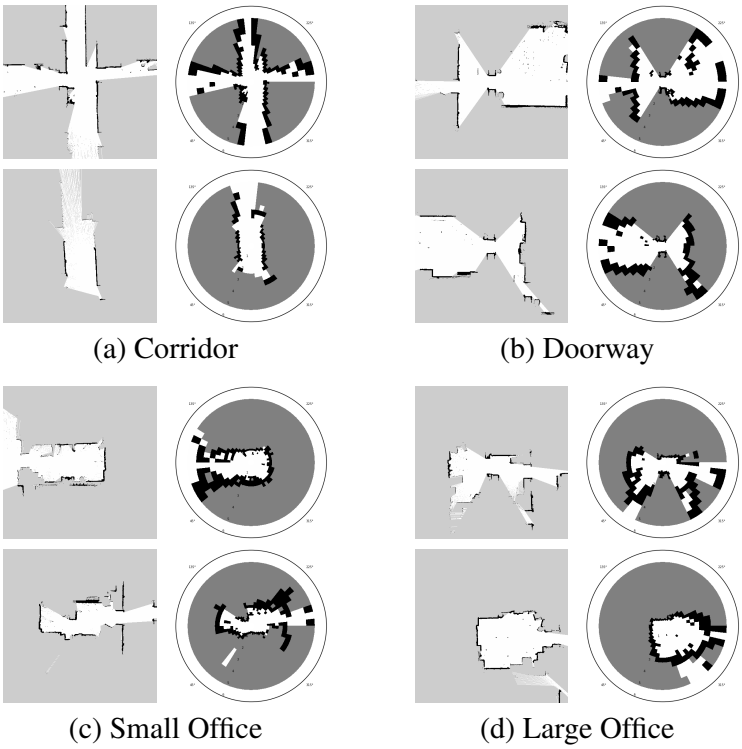


Figure 5.3. Visualization of spatial knowledge represented in the peripersonal layer for sample places of different semantic categories, expressed as both Cartesian and polar occupancy grids.

5.2. Implementation of SK4R$_E$ for Laser-Range Data

In order to evaluate the architecture of SK4R$_E$ in practice, we provide its initial realization for a mobile robot equipped with a laser-range sensor. We utilize laser-range data to simplify the initial implementation, however the proposed representation can be easily extended to include 3D and visual information.

5.2.1. Perceptual Layer

To integrate local laser-range observations in the perceptual layer, we use a common occupancy grid representation. Specifically, we utilized a grid mapping approach based on Rao-Blackwellized particle filters [76]. We crop the resulting grid map to only retain a rectangular fragment of size 10x10m, centered at the current position of the robot. Consequently, we do not require global consistency of the grid map, as long as the local environment is mapped correctly. This will still result in partial maps (especially when the robot enters a new room), but it will help to accumulate observations over time. During our experiments, the robot was exploring the environment driving with a constant speed, while continuously gathering data and performing inferences based on the current state of the perceptual layer.

5.2.2. Peripersonal Layer

The peripersonal representation for each place is constructed from the current local occupancy grid in the perceptual layer. However, since the scope of the peripersonal representation is limited to the space immediately surrounding the robot and relevant context, we only retain information about the parts of the environment visible from the robot (e.g. grid cells that can be raytraced from the robot location). As a result, walls occlude the view and the resulting occupancy grid will mostly contain objects present in a single room. Examples of such local occupancy grids can be seen in Figure 5.3.

Next, every local grid map is transformed into an ego-centric polar representation (compare polar and Cartesian grids in Figure 5.3). This encodes high-resolution information about the geometry and objects nearby, and complements it with less-detailed context further away from the robot. Encoding spatial knowledge closer to the robot in more detail is important for understanding the semantics of the exact robot location (for instance when the robot is in a doorway). However, it also relates to how spatial information is used by a robot when planning and executing actions. It is in the vicinity of the robot that higher accuracy of spatial information is required. The polar grids in our implementation assumed radius of $5m$, with angle step of 6.4 degrees and resolution decreasing with the distance from the robot.

5.2.3. Topological Layer

The topological layer is maintained by a mapping algorithm discretizing continuous space into sets of *places*, *placeholders*, *views*, and *paths*. The goal is to generate an efficient discretization, which supports all the roles of the topological layer, including expression of the global robot pose, representation of affordances related to navigation and exploration, and anchoring of local spatial knowledge to the global space.

The mapping algorithm expands the topological layer incrementally, adding placeholders at neighboring unexplored locations, and connecting them with paths to existing places. Then, once the robot performs an exploration action associated with a specific path, a new place is generated to which a peripersonal representation, as well as place and view descriptors are anchored. At this point, the path between the two places signifies navigation affordance, and is associated with probability based on current, up-to-date information. In order to choose the location for a new placeholder, the algorithm relies upon information contained in the perceptual layer, including detailed local geometry and obstacles.

Similarly to [38], we formulate the problem of finding placeholder locations using a probability distribution that models their relevance and suitability. However, instead of sampling locations of all places in the environment at once, we incrementally add placeholders as the robot explores the environment, within the scope of the perceptual layer. Specifically, the probability distribution is modeled as a combination of two components:

$$P(E \mid G) = \frac{1}{Z} \prod_i \phi_I(E_i)\phi_N(\mathcal{E}), \tag{5.1}$$

where $E_i \in \{0, 1\}$ determines the existence of a place at a location i in the perceptual layer, G is the perceptual occupancy grid, and $\mathcal{E}$ is a set of locations of all existing places within the scope of the perceptual representation.

The potential function ϕ_I models suitability of a specific location, and is defined in terms of three potentials calculated from G:

$$\phi_I(E_i) = \phi_O(E_i)(\phi_V(E_i) + \phi_P(E_i) - \phi_V(E_i)\phi_P(E_i)), \tag{5.2}$$

where:

- ϕ_O ensures that placeholders are created in areas that are safe from collisions with obstacles. It depends on the distance d_o to the nearest obstacle and is calculated similarly to the cost map used on our robot for obstacle avoidance [132]. ϕ_o equals 0 for distance smaller than the radius r of the robot base and $1 - exp(-\alpha(d_o - r))$ otherwise.
- $\phi_V = exp(-\gamma d_c)$ depends on the distance d_c to the nearest node of a Voronoi graph of the 2D map. This promotes centrally located places that are often preferred for navigation.
- ϕ_P promotes places inside narrow passages (e.g. doors). The potential is generated by convolving the local map with a circular 2D filter of a radius corresponding to an average width of a door.

Overall, ϕ_I ensures that placeholders are located only in areas that are safe and preferred for navigation, and constitute useful anchors for information stored in other layers of the representation. The potential ϕ_N, models the neighborhood of a place and guarantees that places are evenly spread throughout the environment. To this end, the potential function promotes positions at a certain distance d_n from existing places:

$$\phi_N(E_i) = \sum_{p \in \mathcal{E}} e^{-\frac{(d(i,p)-d_n)^2}{2\sigma^2}},$$

where $d(i, p)$ is a Euclidean distance between the potential new place and an existing place.

Final location of new placeholders is chosen through MPE inference in $P(E \mid G)$. However, before adding a new placeholder to the map it is important to verify whether the robot will be able to navigate to it. To this end, we perform an A* search directly over the potential function, and quantify the navigability based on the accumulated potential. Only then, a *path* is created between an existing place and a placeholder. Similarly, the accumulated potential is used to quantify navigability of paths between existing places. In order to incorporate knowledge about coarse global geometry into the topological representation, we further relate placeholders and places to a global low-resolution lattice (0.8m distance between points in our experiments), as illustrated in Figure 5.4. As the robot moves through the environment, the lattice is extended, while preserving consistency with existing points. We assume that a place must be associated with a point of the lattice, and each lattice point can be associated with only one place. As a result, when performing MPE inference using $P(E \mid G)$, we assume that only one place might

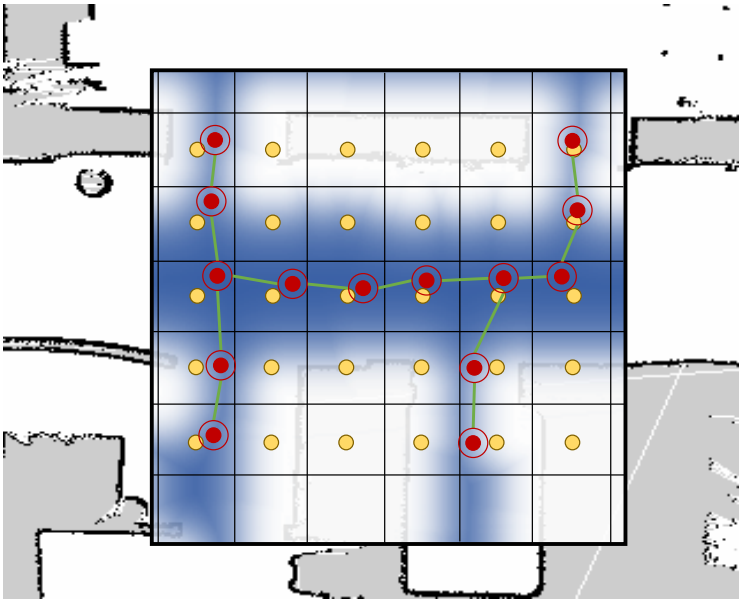


Figure 5.4. Visualization of generated places and paths on top of the knowledge in the perceptual layer. The highlighted region corresponds to the spatial scope of the perceptual representation and displays the value of the potential ϕ_I. The low-resolution lattice is illustrated using yellow points, and red points indicate the final, optimized locations of places. Paths highlighted in green afford navigability throughout the environment.

exist in a cell of a Voronoi tessellation established by the points of the lattice. The resulting set of placeholders (and eventually places) will uniquely correspond to lattice points, yet be created only in locations which are suitable, and can serve as navigation goals for the lower-level controller. For each place that is created from a placeholder, we generate a set of eight *views*. The views are a discrete representation of the heading of the robot when located at a place, and are assumed to be vectors pointing from a point of the lattice to the eight immediately neighboring points. Since, places are associated uniquely with lattice points, each view will naturally point in the direction of only one neighboring place. As a result, each *path* connecting a place to another place or placeholder will be associated with a specific view.

5.2.4. Semantic Layer

In our initial implementation, the semantic layer captures the information about semantic categories of places in the topological map. This includes categories of rooms in which places are located, such as an office or a corridor, but also a functional place category corresponding to places located in a doorway. The layer is implemented as a simple relational data structure assigning place instances to semantic categories in the ontology of the deep default knowledge model. Each such relation is associated with probability value. Additionally, for each place, the layer captures the likelihood of the peripersonal representation of the place being observed for any of the semantic categories. That likelihood is used to detect and explicitly represent that a place belongs to a novel category, i.e. one that is not recognized by the default knowledge model.

5.2.5. Representing Default Knowledge

In our implementation, default knowledge is modeled using a recently proposed Deep Generative Spatial Model (DGSM) [159], a probabilistic deep model which learns a joint distribution over spatial knowledge represented at multiple levels of abstraction For example, DGSM represents the likelihood to associate a table to a kitchen (within the topological layer), or equivalently, the likelihood of a table to have particular geometrical features characterizing a *table* as a category of objects (within the perceptual layer). We apply the deep model to capture generic spatial concepts and relations between knowledge represented in peripersonal, topological, and semantic layers. Once learned, it enables a wide range of probabilistic inferences. First, based on the knowledge in the peripersonal layer, it can infer descriptors of views and places, as well as semantic categories of places. Moreover, it can detect that a place belongs to a novel category, not known during training. Inference can also be performed over the contents of the peripersonal representation. The model can infer missing geometry information resulting from partial observations and generate prototypical peripersonal representations based on semantic information.

To this end, DGSM leverages Sum-Product Networks (SPNs), a novel probabilistic deep architecture [153, 150], and a unique structure matching the hierarchy of representations in SK4R$_E$. Below, we give a primer on Sum-Product Networks and describe the details of the architecture of the DGSM model.

Sum-Product Networks

Sum-product networks are a recently proposed probabilistic deep architecture with several appealing properties and solid theoretical foundations [150, 153, 71]. SPNs represent probability distributions with partition functions that are guaranteed to be tractable, involve a polynomial number of sums and product operations, permitting exact inference. While not all probability distributions can be encoded by polynomial-sized SPNs, recent experiments in several domains show that the class of distributions modeled by SPNs is sufficient for many real-world problems, offering real-time efficiency.

SPNs model a joint or conditional probability distribution and can be learned both generatively [153] and discriminatively [71] using Expectation Maximization (EM) or gradient descent. They are a deep, hierarchical representation, capable of representing context-specific independence. As shown in Figure 5.5 on a simple example of a naive Bayes mixture model, the network is a generalized directed acyclic graph of alternating layers of weighted sum and product nodes. The sum nodes can be seen as mixture models, over components defined using product nodes, with weights of each sum representing mixture priors. The latent variables of such mixtures can be made explicit and their values inferred. This technique is often used for classification models, where the root sum is a mixture of sub-SPNs representing multiple classes. The bottom layers effectively define features reacting to certain values of indicators for the input variables. Not all possible architectures consisting of sums and products will result in a valid probability distribution. However, following simple constraints on the structure of an SPN will guarantee validity (see [153, 150] for details).

Inference in SPNs is accomplished by an upward pass through the network.

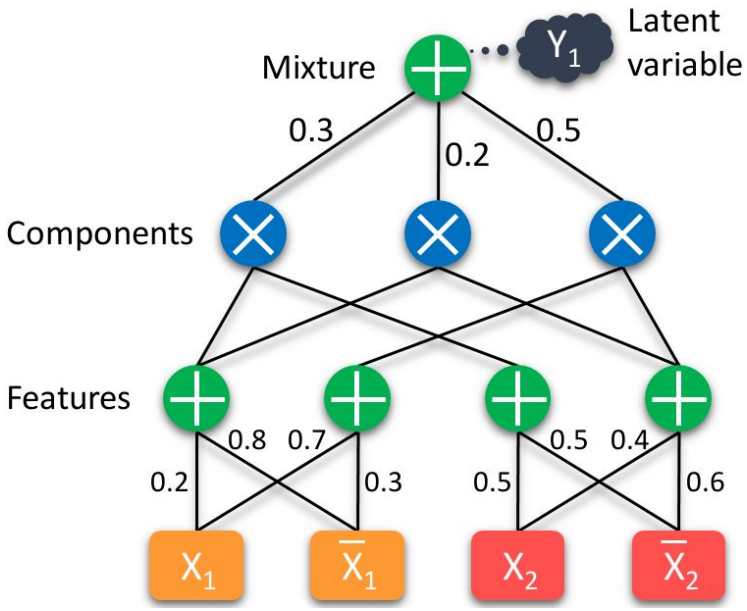


Figure 5.5. An SPN for a naive Bayes mixture model $P(X_1, X_2)$, with three components over two binary variables. The bottom layer consists of indicators for each of the two variables. Weights are attached to inputs of sums. Y_1 represents a latent variable marginalized out by the top sum node.

Once the indicators are set to represent the evidence, the upward pass will yield the probability of the evidence as the value of the root node. Partial evidence (or missing data) can easily be expressed by setting all indicators for a variable to 1. Moreover, it can be shown [153] that MPE inference can be performed by replacing all sum nodes with max nodes. Then, the indicators of the variables for which the MPE state is inferred are all set to 1 and a standard upward pass is performed. A downward pass then follows which recursively selects the highest valued child of each sum (max) node, and all children of a product node. The indicators selected by this process indicate the MPE state of the variables.

In this work, we learn the SPN using hard EM, which was shown to work well for generative learning [153] and overcomes the diminishing gradient problem. Major details of the learning procedure are provided in [159].

Architecture of DGSM

The architecture of DGSM is based on a generative SPN illustrated in Figure 5.6. The model learns a probability distribution $P(C, D_1^P, \ldots, D_{N_p}^P, D_1^{V_1}, \ldots, D_{N_v}^{V_8}, X_1, \ldots, X_{N_x})$, where C represents the semantic category of a place, $D_1^P, \ldots, D_{N_p}^P$ constitute an internal descriptor of the place, $D_1^{V_1}, \ldots, D_{N_v}^{V_8}$ are descriptors of eight views, and $X_1, \ldots, X_C$ are input variables representing the occupancy in each cell of the polar grid of the peripersonal layer. Each occupancy cell is represented by three indicators in the SPN (for empty, occupied and unknown space). These indicators constitute the bottom of the network (orange nodes).

The structure of the model, as in [160] is randomly generated with a random

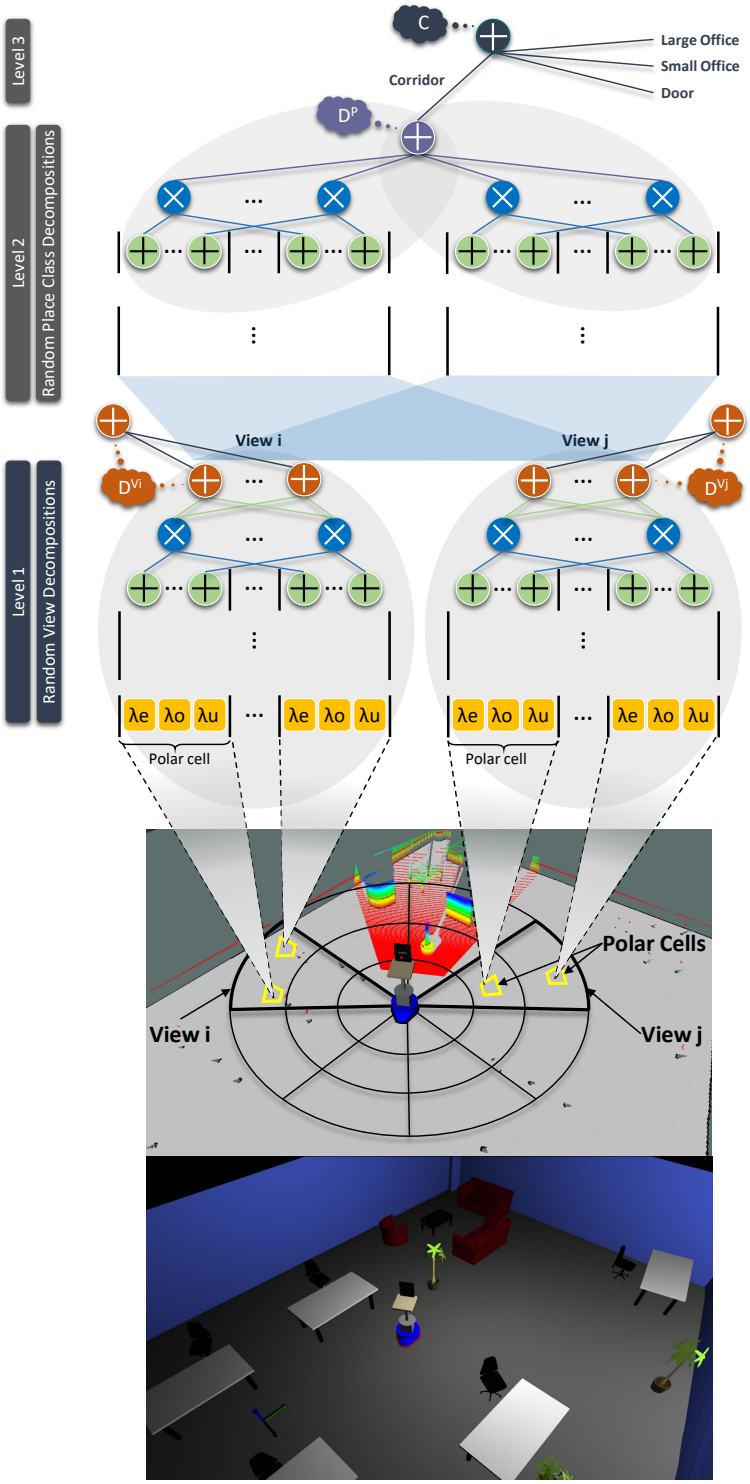


Figure 5.6. The structure of the SPN implementing our spatial model. The bottom images illustrate a robot in an environment and a robocentric polar grid formed around the robot. The SPN is built on top of the variables representing the occupancy in the polar grid.

forest approach. The advantage of such an approach is its ability to represent a remarkable number of probability distributions. In particular, the random structure is generated by recursively re-partitioning the random variables, and associating new partitions to nodes of the SPN. The process ends when each partition contains only a single variable. Moreover, to simplify the SPN structure, if randomly generated branches are not meaningful to the modeled distribution, they can be pruned after parameters learning.

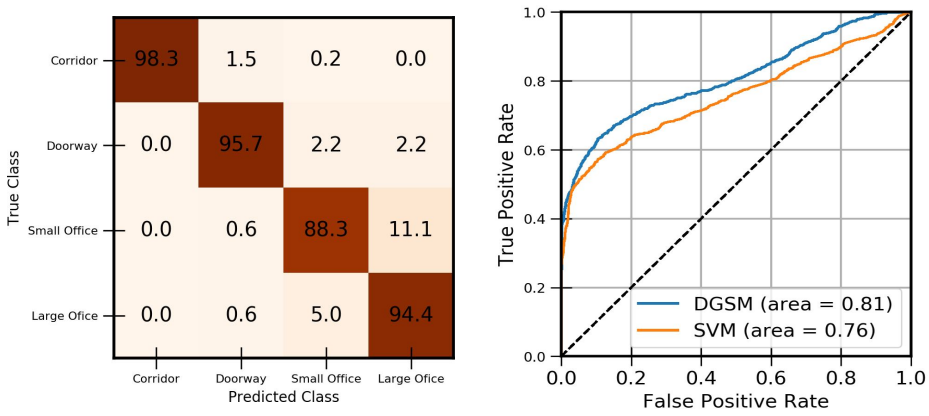
In this work, the resulting model is a single SPN, which is assembled from three levels of sub-SPNs. First, we begin by splitting the polar grid of the peripersonal layer equally into eight 45 degree parts, corresponding to the *views* defined in the topological layer. For each view, we randomly generate a sub-SPN over the subset of X_i representing the occupancy within the view, as well as latent variables $D_1^{V_i}, \ldots, D_{N_v}^{V_i}$ serving as an internal view descriptor. The sub-SPN can be seen as a mixture model consisting of 14 components in our implementation. In the second level, we use the distributions defining the components from each view ($8*14$ in total) as inputs, and generate random SPNs representing each of the semantic place classes in the ontology. Each of such SPNs is itself a mixture model with the latent variable D_i^P being part of the place descriptor. Finally, in the third level, the sub-SPNs for place classes are combined by a sum node (mixture) forming the root of the whole network. The latent variable associated with the root node is C and is set to the appropriate class label during learning. Overall, such decomposition allows us to use networks of different complexity for representing lower-level features of each view and for modeling the top composition of views into place classes.

5.3. Experimental Evaluation

Our experimental evaluation consists of two parts. First, we evaluated the ability of the deep default knowledge model implemented with DGSM to perform both top-down and bottom-up inferences across the layers of the representation. Then, we deployed our complete implementation of SK4R$_E$ in order to build representations of large-scale environments.

5.3.1. Experimental Setup

Our experiments were performed on laser range data from the COLD-Stockholm database [155]. The database contains multiple data sequences captured using a mobile robot navigating with constant speed through four different floors of an office building. On each floor, the robot navigates through rooms of different semantic categories. Four of the room categories contain multiple room instances, evenly distributed across floors. There are 9 different *large offices*, 8 different *small offices*, 4 long *corridors* (1 per floor, with varying appearance in different parts), and multiple examples of observations captured when the robot was moving through *doorways*. The dataset features several other room categories: an elevator, a living room, a meeting room, a large meeting room, and a kitchen. However, with only one or two room instances in each. Therefore, we decided to use the four categories with multiple room instances for the majority of the experiments and designated the remaining classes as novel when testing novelty detection.



(a) Confusion Matrix

(b) Novelty ROC curve

Figure 5.7. Results of experiments with bottom-up inference: (a) normalized confusion matrices for semantic place categorization; (b) ROC curves for novelty detection (inliers are considered positive, while novel samples are negative).

To ensure variability between the training and testing sets, we split the samples from the four room categories four times, each time training the model on samples from three floors and leaving one floor out for testing. The presented results are averaged over the four splits.

5.3.2. Bottom-up Inference

First, we evaluated the ability of DGSM to infer semantic place categories given information in the peripersonal layer. As a comparison, we used a well-established model based on an SVM and geometric features [136, 156]. The features were extracted from laser scans raytraced in the same local Cartesian grid maps used to form polar grids of the peripersonal layer. We raytraced the scans in high-resolution maps (2cm/pixel), to obtain 362 beams around the robot. To ensure the best SVM result, we used an RBF kernel and selected the kernel and learning parameters directly on the test sets. The models were trained with peripersonal representations obtained for locations on three floors in places belonging to four place categories, and evaluated on the fourth floor or using data from rooms designated as novel. The classification rate averaged over all classes (giving equal importance to each class) and data splits was $85.9\% \pm 5.4$ for SVM and $92.7\% \pm 6.2$ for DGSM, with DGSM outperforming SVM for every split. The normalized confusion matrix for DGSM is shown in Figure 5.7a. Most of the confusion exists between the small and large office classes. Offices in the dataset often have complex geometry that varies greatly between room instances.

Additionally, we evaluated the quality of the uncertainty measure produced by DGSM and its applicability to detecting novel concepts. To this end, we thresholded the likelihood of the test peripersonal representations produced by DGSM to decide whether the robot is located in a place belonging to a class known during training. We compared to a one-class SVM with an RBF kernel trained on the geometric features. The cumulative ROC curve for the novelty detection experiments over all

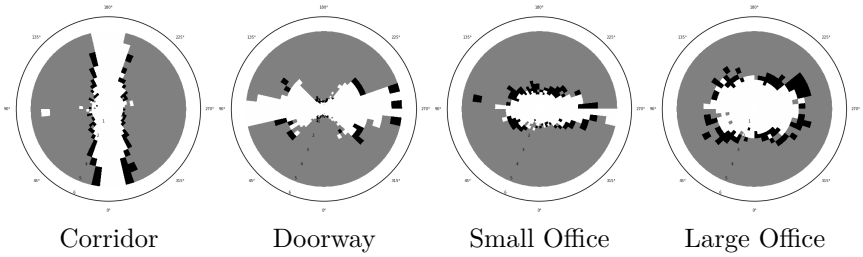


Figure 5.8. Prototypical peripersonal representations inferred from semantic place category.

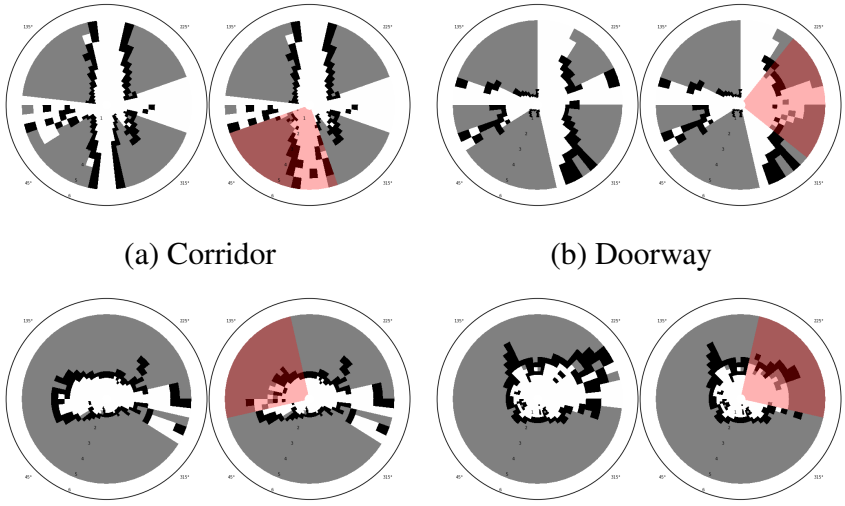


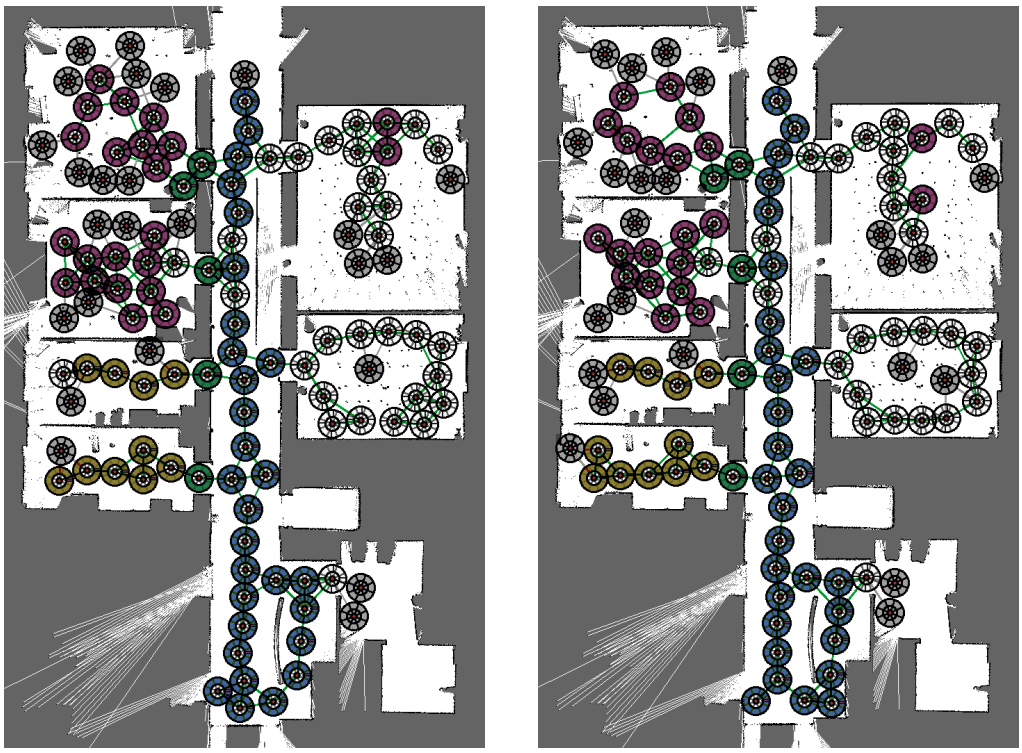
Figure 5.9. Examples of completions of peripersonal representations with missing data grouped by true semantic category.

data splits is shown in Figure 5.7b. We see that DGSM offers a significantly more reliable novelty signal, with AUC of 0.81 compared to 0.76 for SVM.

5.3.3. Top-down Inference

In the second experiment, we used DGSM to perform inference in the opposite direction, and infer values of cells in the peripersonal representation. First, we inferred complete, prototypical peripersonal representations of places knowing only place semantic categories. The generated polar occupancy grids are shown in Figure 5.8a-d. We can compare the plots to the true examples depicted in Figure 5.3. We can see that each polar grid is very characteristic of the class from which it was generated. The corridor is an elongated structure with walls on either side, and the doorway is depicted as a narrow structure with empty space on both sides. Despite the fact that, as shown in Figure 5.3, large variability exists between the instances of offices within the same category, the generated observations of small and large offices clearly indicate a distinctive size and shape.

Then, we used DGSM to generate missing values in partial observations of places. To this end, we masked a random 90-degree view in each test polar grid



(a) Run #1. **(b)** Run #2.

Figure 5.10. Contents of the topological and semantic layers after two different runs over 5-*th* floor. Gray nodes represent placeholders, while blank nodes indicate places detected as belonging to novel categories. Colors indicate recognized semantic place categories: blue for a corridor, green for a doorway, yellow for a small office, and magenta for a large office. The two large bottom rooms belong to a novel category: "meeting room".

(25% of the grid cells). All indicators for the masked polar cells were set to 1 to indicate missing evidence and MPE inference followed. Figure 5.9 shows examples of peripersonal representations filled with predicted information to replace the missing values. Overall, when averaged over all test examples and data splits, DGSM correctly reconstructed $77.14\% \pm 1.04$ of masked cells. This demonstrates its generative potential.

5.3.4. Representing Large-Scale Space

In our final experiment, we deployed the complete implementation of SK4R$_E$ and evaluated its ability to build comprehensive, multi-layered representations of large-scale space. Specifically, we tasked it with representing the 5-*th* and 7-*th* floor of the office building in the COLD-dataset, which measure respectively 298 and 435 square meters. In each case, we incrementally built the representation based on the sensory data captured as the robot navigated through the environment. We relied on the perceptual layer to perform low-level integration of observed laser scans, on peripersonal layer to capture local place information, the topological layer to maintain a consistent topological graph expressing navigability and knowledge gaps related to unexplored space, and finally on the semantic layer to encode information

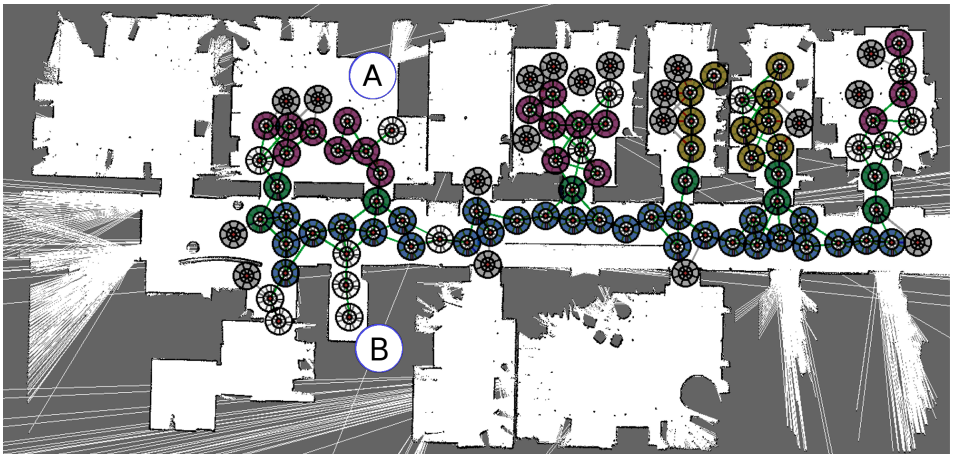


Figure 5.11. Contents of the topological and semantic layers after a single run over the 7-*th* floor. Gray nodes represent placeholders, while blank nodes indicate places detected as belonging to novel categories. Colors indicate recognized semantic place categories: blue for a corridor, green for a doorway, yellow for a small office, and magenta for a large office. The rooms marked with letters A and B belong to novel categories: "living-room" and "elevator".

about semantic categories of places, including detections of novel semantic categories.

Figure 5.10 illustrates the state of the representation after two completed runs over the 5-*th* floor. The figure presents the final topological graph of places visited by the robot, paths expressing navigability between them, as well as paths leading to placeholders representing possibility of further exploration. For each place, we use color to illustrate the inferred semantic category, or detection of a novel category. First, we can observe that places are evenly distributed across the environment and exist in locations which are relevant for navigation or significant due to their semantics (e.g. in doorways). Moreover, the graphs created during different runs are similar and largely consistent. Second, the semantic place categories inferred by DGSM agree with the ground truth when the category of the place was recognized as known. To detect novel classes, we again thresholded the estimates of the likelihood of the peripersonal representations provided by DGSM. On the 5-*th* floor, the novel category was "meeting room" and two meeting rooms are shown in the bottom part of the map. Although both false positives and false negatives exist, places in both meeting rooms are largely correctly classified as belonging to novel categories.

Figure 5.11 shows results for a different environment, the 7-*th* floor. Similar observations can be made as for the 5-*th* floor. However, here the novelty detection is less accurate. DGSM correctly detects the places in the elevator as novel (marked with "B" in the figure), but it fails to detect novelty in the living room ("A" in the figure), which instead is misclassified as "large office". While not a desirable outcome, it is not surprising, given the similarity between the living room and large offices in the dataset when observed solely using laser range sensors.

5.4. Concluding Remarks

We center the study in this chapter around an important aspect of the SK4R representation. We investigated whether it is possible to formalize SK4R$_E$ to support action planning by taking in exam the following question

1. *Is it possible to define a domain-specific representation of the environment that allows the robot to encode affordance semantics at different levels of abstraction of spatial knowledge?*

Our solution gave us the opportunity to better understand how to structure the state of complex environments for an autonomous robot. A suitable state representation is crucial to make the robot aware of its physical surroundings and guarantee task completion. Hence, we formalized SK4R$_E$ to provide a practical state of the operational environmental s_t to the robot, by representing meaningful spatial landmarks across different levels of abstractions.

To this end, we designed SK4R$_E$ specifically to represent the belief about the state of the world and encode spatial affordances on a mobile robot. We demonstrated that an implementation following the principles described in this chapter can successfully represent general spatial concepts at multiple levels of abstraction, and utilize them to obtain a complete and comprehensive model of the robot environment E – even for a relatively simple sensory input. The experimental evaluation shows that SK4R$_E$ can effectively label particular areas of the environment and infer missing knowledge by performing top-down inference. Then, it is able to encode the state of large environments in order to support action execution of indoor service robots. It is important remarking that the SK4R$_E$ architecture supports planning at different levels. It enables long-term action planning and epistemic actions, and also, more reactive behaviors in the peripersonal domain of the robot.

Finally, the current implementation of SK4R$_E$ can be further improved in different ways. For example, we can include more complex perceptions provided by visual and depth sensors. Additionally, we can train the deep model of default knowledge to directly predict complex place affordances related to human-robot interaction.

Chapter 6

SK4R for Social Interactions

Operating in *human-populated* environments is a key and a very challenging task for robots. They, in fact, have to understand and interpret social cues in a human compatible manner. On the one hand, social behaviors are mandatory to enable human-robot cooperation. The social configuration of the environment is an important component of the state of the robot. This is especially important in applications running in the context of service robotics, education and health-care. On the other hand, the "social state" of the environment is not usually observable through robot sensors and needs to be encoded in an initial representation of the robot beliefs. To this end, we dedicate this chapter to the study of environmental *factors* enabling a more natural social interaction among human and robots. In particular, we investigate (i) how to represent social features in SK4R$_E$, (ii) how to design those features and discuss (iii) how the robot should behave accordingly. Section 6.1 contextualizes our work and formalizes its underlying concepts. Then, Section 6.2 analyzes the methodology used to collect knowledge about social components of the environment and reports the experimental evaluation of two user studies conducted in our department. Finally, Section 6.3 summarizes the chapter and discusses its findings.

6.1. SK4R$_E$ for Social Interactions

In social scenarios, robots are expected to cooperate with humans and, therefore, to interact with them by showing safe and acceptable behaviors [101, 164]. Hence, as an active component of the state of the environment $s_t \in S$, robots have to be able to understand their social working context. Our goal, in this chapter, is to include social factors into the robot understanding of the world, and then let it shape its behaviors upon selected social cues. More precisely, we extend the problem of characterizing the state of the environment $s_t \in S$, for a κ-agent, to include social components in its belief. As in the previous chapter, we focus on SK4R$_E$, the environmental module of SK4R (see Figure 5.1, Page 62), and we extend the state of the environment to include *social factors* that we analyze through two user studies.

In particular, we study social factors in human-robot interactions where the embodiment of the robot is not sufficient to execute tasks usually carried out by

humans. For example, as a variety of robots (especially low-cost mobile robots) do not feature a manipulator, they are not able to grasp an object, open a door or simply push a button. In literature, to overcome limitations due to the robot embodiment (and succeed in these kind of tasks), the research community has introduced and investigated the concept of *Symbiotic Autonomy* [183] or *Symbiotic Robotics* [41], where robots perform service tasks for humans, while humans help them to achieve their goals. Generally in Human-Robot Interaction (HRI) studies, research addresses the case of humans asking for help [63, 141]. In such a scenario, interactions are triggered by humans in order to take advantage of robots' services. Instead, we study the configuration of a different social context, that wants to evaluate which are the component of s_t that (1) allow the robot to ask humans' help and (2) maximize its likelihood to be helped. By relying upon previous work [174, 1], we study this particular configuration of the Symbiotic Autonomy through two user studies. Such an evaluation of social components – that extend the social understanding of s_t of the robot – constitute the major contribution of this chapter. In particular, we analyze which factors influence human attitude to help the robot. We hypothesize that such attitude has not a constant value as it depends on several factors imposed by human physiology and by the context in which they are currently in – i.e. elements of s_t. To this end, we introduce the novel concept of *Collaboration Attitude* (CA) as a quantitative measure to characterize such an inclination. To evaluate CA, we conduct two independent user studies with the aim to socially characterize the state of the environment, and thus, to generate behavioral guidelines for social robots. In both user studies, we take into exam specific environmental components that influence CA. In the former study we analyze "Proxemics" (i.e., relative pose of the interactive partners), "Gender" and "Height" of the experimenters and "Context" (i.e., operational environment of the interaction). While, in the latter, we investigate how the effects of the "Activity" that the experimenter is performing, influences their attitude in helping the robot.

Throughout this chapter we provide a formalization of Collaboration Attitude and define our working hypotheses for the two user studies. Then, we present (i) our system, (ii) the setup of the experiments and (iii) the statistical evaluation of the collected results. Finally, we conclude the chapter by discussing the findings of our study and proposing new factors that might aid the representation of social environmental features into the state of SK4R.

6.2. Collaboration Attitude

In order to highlight important components of the environment state $s_t \in S$ that can influence the behavior of the robot, we propose the study and evaluation of the Collaboration Attitude (CA). CA, in fact, framed in the context of Symbiotic Autonomy [182], aims at highlighting environmental social components of the environment to be included in the state of SK4Rs_t. In particular, we characterize features of s_t that may or may not favor humans' attitude in helping the robot. Hence, to study the collaborative inclination of humans toward robots, we need a quantitative measure that captures the concept of Collaboration Attitude. To this end, the Collaboration Attitude has been modeled as a N-point Likert scale as

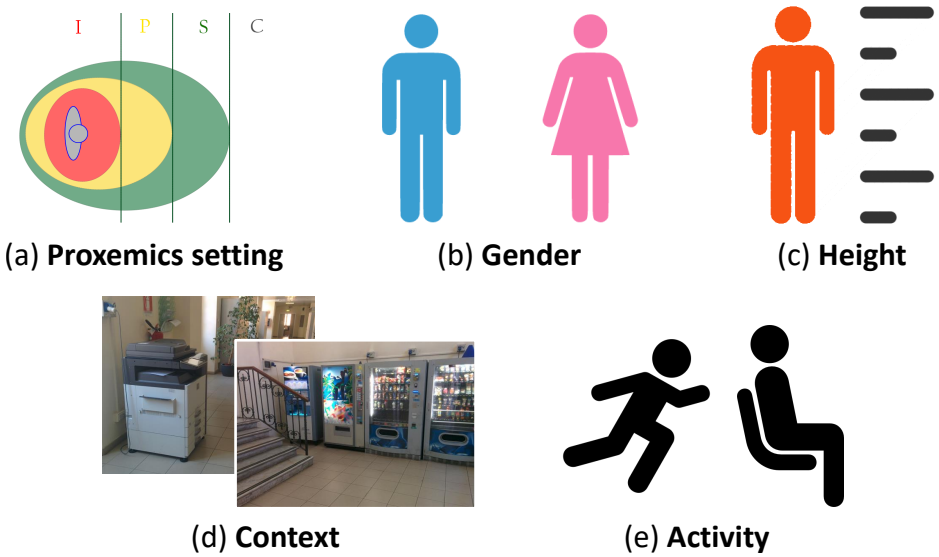


Figure 6.1. Social factor analyzed to contextualize the social configuration of the state of the environment. Top to bottom and from left to right, the factors are "Proxemics Settings"(a), "Gender"(b), "Height"(c), "Context"(d) and "Activity"(e).

follows:

Definition 6.1 (Collaboration Attitude)**.** *The Collaboration Attitude measures the attitude of humans toward the requests for help of the robot in a Symbiotic Autonomy framework. Formally, it is quantified according to metrics defined on a scale of N points, where N is the number of tasks that the human is requested to accomplish. Precisely, the Collaboration Attitude assumes values in $[0, \ldots, N-1]$, where 0 represents lowest level of collaboration, i.e., the human is not willing to help at all, while $N-1$ represents the highest one, i.e., the human is willing to help the robot in all the tasks.*

Accordingly, we formalize our working hypotheses of the two user studies. The hypotheses are formalized in order to highlight possible factors (i.e. components of $s_t \in S$ illustrated in Figure 6.1) that influence the CA of humans when a human-robot social interaction is conducted. In particular, we focus on "Proxemics Settings"(a), "Gender"(b), "Height"(c), "Context"(d) and "Activity"(e). "Proxemics Settings" models the relative distance and orientation of the two interactive partners. "Gender" and "Height", instead, are attributes of the experimenters. Conversely, "Context" and "Activity" model factors relating to the situation in which the interaction happen. The former highlights the type the scenario of interaction, whether it is carried out in a relaxing or working context (e.g. vending machine vs. library). The latter, instead, pinpoints the activity performed by the user before robot interruption. Whether the user was standing and drinking a coffee or sitting studying for a class/exam.

6.2.1. User Study 1: Proxemics, Gender, Height and Context

In this user study, the robot asks people for help in different Contexts (namely, *Relaxing* and *Working*), with different Proxemics settings (namely, *Intimate*, *Personal*

and *Social*). The analysis of such factors generates a model of interaction that defines: (i) whether they actually influence the Collaboration Attitude, and (ii) the values that maximize it. In particular, we analyze four hypotheses which relate to the factors illustrated in Figure 6.1.

Definition 6.2. *Collaboration Attitude is subject to different Proxemics settings*

It is well known that among humans and robots, Proxemics has a key role in the interaction. Therefore, experiments aim at highlighting the importance of respecting the personal space in social interactions, even in the case where the interactive partner is a robot. Specifically, we want to estimate whether different settings of Proxemics (see Figure 6.1a) might vary the Collaboration Attitude that the human shows, ranging from an *intimate* distance to a *social* one.

Definition 6.3. *Collaboration Attitude is subject to the gender of the human*

Humans' physical and social characteristics affect how they behave in different situations. Gender is one of the major features to be considered. Such factor is usually considered in HRI studies, as males and females show different responses to equal stimuli.

Definition 6.4. *Collaboration Attitude is subject to the height of the human*

Robot appearance – and more precisely, humans' perception of the robot – constitutes a key factor to be investigated, when studying humans response to robot behaviors. Our intuition is that shorter people perceive the robot differently than taller people and their Collaboration Attitude varies depending on such a perception. The outcomes of the statistical analysis over the collected data confirm that the Collaboration Attitude has a not constant value when different factors are changing.

Definition 6.5. *Collaboration Attitude is subject to different Contexts*

The environmental context of the interaction plays a central part in social interactions. Humans behave differently, depending on where they are and the contexts they are in. Consequently, a robot needs to consider these social elements.

6.2.2. User Study 2: Human Activity

In this second user study, we setup a robot that interrupts people and asks for help. The agent approaches people that are involved in different activities and thus might respond to the evaluation differently. We are interested in analyzing the following hypothesis.

Definition 6.6. *Collaboration Attitude is subject to different human activities*

The activity in which the person is involved when the robot asks for help affects the level of collaboration toward the human. Specifically, we consider users to be in a *Standing* activity, if they stand at a location or are walking – for example, whenever they are going to a meeting or attending a class, or equivalently, if they are having a coffee. We consider users to be in the *Sitting* activity, instead, if they are sitting in the open areas, for example taking a break, having lunch or studying.

Figure 6.2. Modified Turtlebot robot. The platform deployed is higher than the standart version, and features a tablet which is used to carry out interactions with users.

6.2.3. User Study Methodology

The degree of Collaboration Attitude in changes to the dependent factors has been analyzed through two subsequent user studies. Moreover, to guarantee validity of the experimental evaluation, the subjects selection policy, apparatus, procedure and questionnaire have been preserved. For each user study, we executed different runs of the same experiment by interrupting users in different contexts and activities by varying our population according to the studied factors. This section (1) introduces our subject population, and (2) provides a detailed description of the tools and the procedure used into the experimentation. Accordingly, it presents the questionnaire used to collect users' data.

Subjects

All the experiments have been conducted in a department of our university. In such an environment, the users have been randomly selected from a set of students with homogeneous characteristics, all of them between 20 and 30 years old. Moreover, in both user studies, the experiment is completed in a *"between group" design*, so that every user participated only once and the data collected is not biased by repetitions of the experiments by the same user. Participants have not been compensated, nor have they provided any consensus for taking part to the experiment. Otherwise, either a consensus or a reward, would have biased the attitude of the users and invalidate the whole experiment.

Apparatus

In both user studies, the deployed the same modified version of the Turtlebot Robot (see Figure 6.2). While the base remains unaltered, the structure on top of it has been customized, in order to make the robot taller with respect to the standard version. In fact, it is 98 cm high and it features a tablet on top as an interface for spoken interactions. We allow users to have *short-term dialogues* with the robot, to support the estimation of the attitude of the human to help the robot in performing its tasks. Our short-term dialogue system is composed of two main modules: (i) an *Automatic Speech Recognizer* (ASR), that processes the acoustic signal of the users' speech and generates a set of possible transcriptions; (ii) a *Dialogue Manager* (DM) that manages the dialogic interaction. The ASR module has been realized through the Google Speech APIs, available within the Android environment, in an ad-hoc mobile application. The app is also in charge of managing the questionnaire presented to the user at the end of the interaction, through a touch-based Graphical User Interface (GUI). The dialogue flow is managed through an Artificial Intelligence Markup Language (AIML) Knowledge Base.

Procedure

We conducted our studies both in closed and open areas of a department in our university, where the heterogeneity of both environment and population gives the opportunity to collect data for each value of the considered factors. The whole experiment is conducted in a Wizard-of-Oz fashion [180] and includes a predefined set of four phases, namely *Approach*, *Dialogue*, *Questionnaire* and *Homing*. During the *Approach* phase, the robot approaches the user that is not aware of being involved in the study until the questionnaire is displayed. Given the purpose of the study, only this phase slightly differs depending on the factors and their value. In fact, once we select the next user, we let the robot notify its presence and seek for help. The robot asks the experimenter to keep his/her position. Afterwards, the robot approaches the user within the "Personal" Proxemics setting – kept constant for each user. We did not vary the orientation of the robot during the experiments, as other works [104, 217] focused on the relative orientation of the robot with respect to the user. In the case of the sitting activity, the robot goes towards the user by respecting the same social distance [78, 217], while in the standing setting, it intercepts the human which is passing by or standing still. After that the user attention is gained, the *Dialogue* phase is triggered and the robot asks to be helped in a particular task. After this short interaction, the robot displays the *Questionnaire* on the table aiming at completing the evaluation of CA and collecting users' information. Once the questionnaire has been completely filled in, the *Homing* phase is executed, where the robot thanks the user and is guided toward its original position. It is worth emphasizing that the chosen characterizations of Context and Activity are done by taking into account the actual abilities of the robot perception.

Questionnaire

During the two user studies, we collected, with the same methodology, data by asking the user to fill in a questionnaire that the robot displays on the tablet. We divided

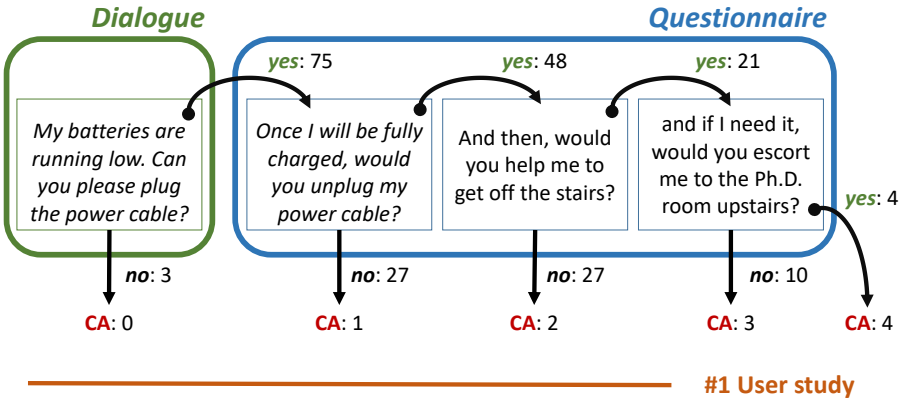


Figure 6.3. Collaboration Attitude estimation through questionnaire for the first User study. Users'answers are highlighted in the figure.

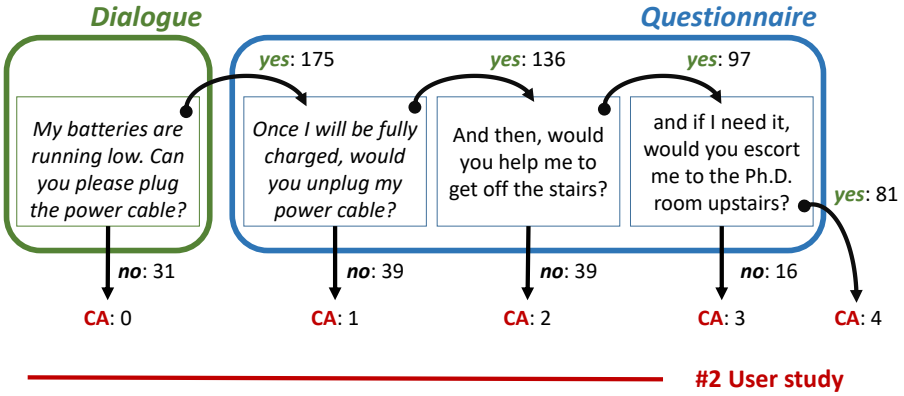
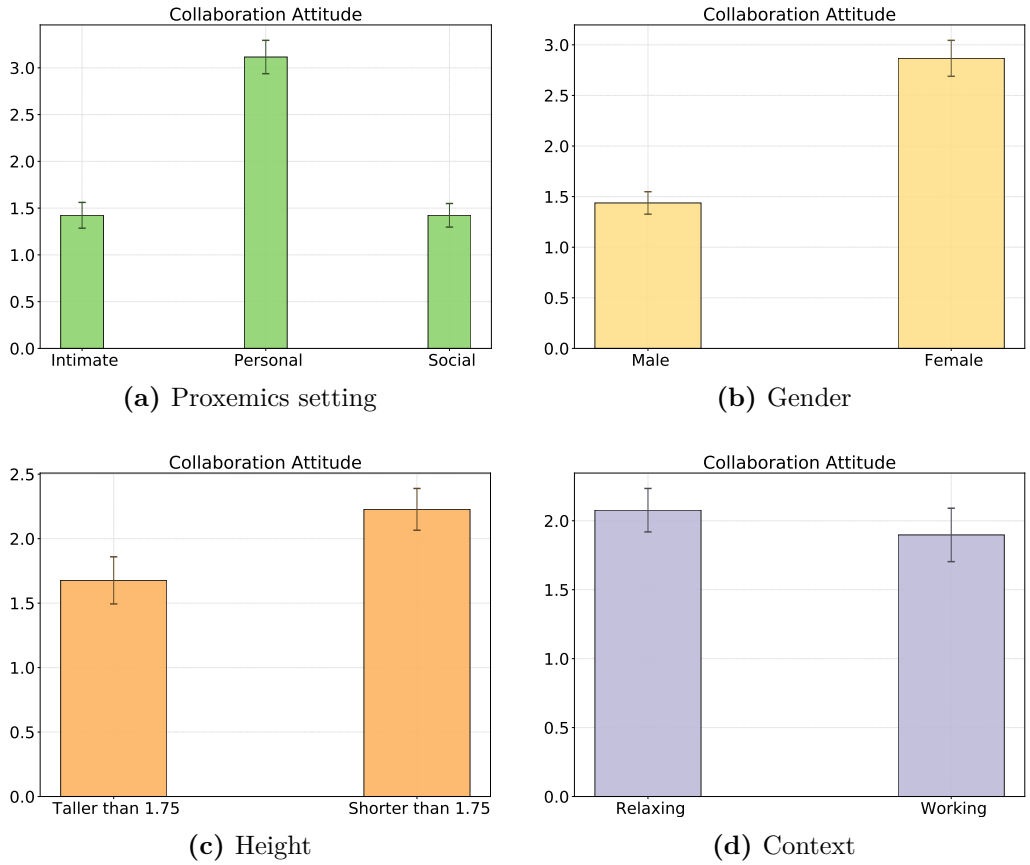


Figure 6.4. Collaboration Attitude estimation through questionnaire for the second User study. Users'answers are highlighted in the figure.

the questionnaire into two sections aiming at (i) quantifying the Collaboration Attitude, and (ii) collecting information about the user. Specifically, we characterize users by gathering information about gender, height and acquaintance toward robotics. The Collaboration Attitude is mapped into a 5-point scale, measuring the amount of positive responses of the experimenters to the robot requests, according to Definition 6.1. Hence, if we consider also the initial request (in the *Dialogue* phase), this variable takes values in $\{0, ..., 4\}$, where 0 is the case where the human is not willing to help the robot in any task and 4 the opposite situation. Figure 6.3 and Figure 6.4 show the requests posed to the experimenters. While the first request is part of the dialogic interaction, the remaining three are both uttered by the robot and displayed as part of the questionnaire. In both figures, the number on each edge refers to the occurrences of a particular answer of the second user study, i.e., *yes* or *no*. In particular, arcs labelled with *no* represent users giving up in helping the robot at a particular CA request, while arcs labelled with *yes* count users that advanced through the different questions. For instance, in the second user study, the 31 users neglecting the initial request achieved a CA of 0, while the 81 users – that satisfied all the robot requests – obtained a CA score of 4. As one might expect, in both user studies, the engagement decreases as the requests become more and



(a) Proxemics setting **(b)** Gender

(c) Height **(d)** Context

Figure 6.5. Collaboration Attitude means and standard errors of the first user study [174]

more demanding. In fact, while in the second study, at each question (except for the fourth request), 36 users on average abandoned the robot, for the first user study this is even more remarked. In fact, only 4 users positively reached the last stage.

6.2.4. User Study 1: Experimental Results

This section reports the results obtained in the first user study [174]. In Figure 6.5, the Collaboration Attitude means and standard errors are plotted.

The Proxemics setting that maximizes the Collaboration Attitude is when the robot approaches the human with a Personal distance (Figure 6.5a). This result is in line with other user studies conducted in Human-Robot Proxemics [212, 137, 133], stating that humans' comfort is maximized within the Personal setting. The Intimate and Social distances give lower values of Collaboration Attitude.

When looking at the gender of the experimenters (Figure 6.5b), the mean of the Collaboration Attitude obtained by females is strikingly higher than the males' one. This represents a first indication that females are more inclined to help robots than males. The study of this factor is interesting, as it is known that males and females have different social behaviors.

Conversely, Figure 6.5c shows statistics of the Collaboration Attitude means to changes in height of the experimenters. It seems that shorter experimenters are

more inclined in collaborating with respect to taller ones.

Despite the Relaxing context seems to maximize the collaborative intentions of the experimenters (Figure 6.5d), the Collaboration Attitude is rather stable when different contexts are tested. As a consequence, the Context does not appear to be a perturbing factor for the Collaboration Attitude.

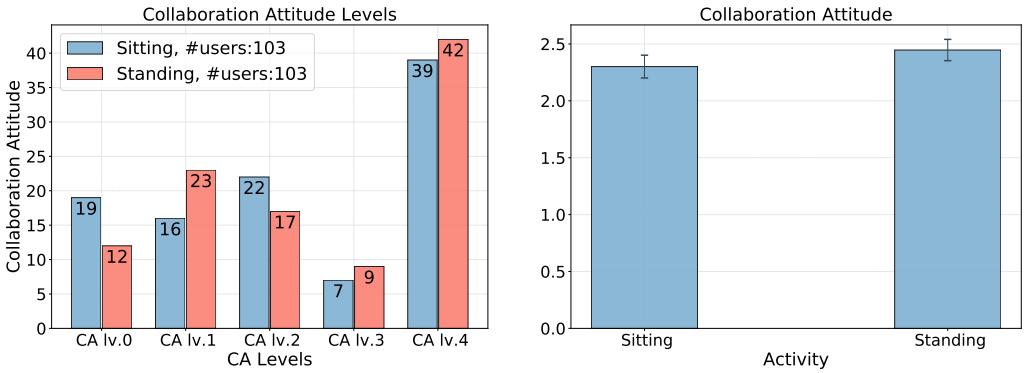
In order to search for significant variations and test our operational hypotheses, we performed One-Way ANOVA over the different datasets. In the left part of Table 6.1, a sketch of the sample under consideration is shown. The populations of Proxemics and Context factors are completely balanced, with a population of 26 elements for each Proxemics setting and 39 experimenters for both the Relaxing and Working groups. Conversely, the samples of the Gender factor are not balanced, with a majority of males with respect to females, i.e., 62% vs. 38% and a prevalence of shorter experimenters, i.e., 56% shorter vs. 44% taller.

Table 6.1. One-Way ANOVA results

Groups	*Count*	*Sum*	*Avg*	*Var*
Intimate	26	37	1.42	0.49
Personal	26	81	3.12	0.83
Social	26	37	1.42	0.41
Male	48	69	1.44	0.59
Female	30	86	2.87	0.95
Taller 1.75m	34	57	1.68	1.13
Shorter 1.75m	44	98	2.23	1.16
Relaxing	39	81	2.08	0.97
Working	39	74	1.9	1.46

Src of Var	*SS*	*df*	*MS*	*F*	*p-val*	*F crit*
			Proxemics			
Btw. Groups	49.64	2	24.82	42.95	$3.71 \cdot 10^{-13}$	3.12
Wtn. Groups	43.35	75	0.58			
Total	92.99	77				
			Gender			
Btw. Groups	37.71	1	37.71	51.84	$3.7 \cdot 10^{-13}$	3.967
Wtn. Groups	55.28	76	0.73			
Total	92.99	77				
			Height			
Btw. Groups	5.82	1	5.82	5.07	0.027	3.97
Wtn. Groups	87.17	76	1.15			
Total	92.99	77				
			Context			
Btw. Groups	0.63	1	0.63	0.52	0.47	3.97
Wtn. Groups	92.36	76	1.22			
Total	92.99	77				

The right part of Table 6.1 shows the ANOVA results by reporting the *P-value*, the *sum of squares* (*SS*), the *degrees of freedom* (*df*), the *mean squares* (*MS*), the *ratio of the two mean squares values* (*F*) and the *F critical value* (*F crit*). The



(a) Users' Collaboration Attitude levels divided with respect to the "Activity" factor

(b) Collaboration Attitude means and standard errors with respect to the "Activity factor"

Figure 6.6. Collaboration Attitude analysis of the second user study

Collaboration Attitude depends on the Proxemics setting chosen for the experiment (p-value < 0.05). In order to confirm the ANOVA results, we performed a post-hoc test through three t-tests, aimed at comparing each pair of groups. Table 6.2 shows

Table 6.2. t-Test: Two-Sample Assuming Equal Variances

	Intimate vs. *Personal*	*Intimate* vs. *Social*	*Personal* vs *Social*
df	50	50	50
P(T<=t) two-tail	$9.6 \cdot 10^{-10}$	1	$4.1 \cdot 10^{-10}$

the result of this additional analysis. As suggested by the means histogram, in the Personal distance humans act differently w.r.t. Intimate and Social settings (the *two-tailed p* values are lower than 0.05), whereas users seem to behave similarly in their Intimate and Social spaces. Also the Gender seems to be a significant factor for the Collaboration Attitude. In fact, in the right part, the One-Way ANOVA results allow to reject the null hypothesis (p-value < 0.05). By looking at the results, there is a significant variation among the different clusters of Height. In fact, the p-value is lower than 0.05 and we reject the null hypothesis with 95% of confidence. Regarding the Context, the outcomes confirm our previous observations, with a p-value significantly greater than 0.05.

6.2.5. User Study 2: Experimental Results

In this section, we report and discuss the results of the second user study. Again, in order to analyze the collected data, we performed a One-Way ANOVA test. Here we focus on a more fine-grained discretization of the "Activity" that users are performing at the moment of the interaction.

Figure 6.6a shows the number of users grouped with respect to the CA value that they achieve and divided according to the two values of the activity factor. Figure 6.6b, instead, reports means and standard errors of the Collaboration Attitude.

Table 6.3. Activity: One-Way ANOVA results

Groups	*Count*	*Sum*	*Avg*	*Var*
Sitting	103	237	2.30	1.54
Standing	103	252	2.45	1.49

Src of Var	*SS*	*df*	*MS*	*F*	*p-val*	*F crit*
			Activity			
Btw. Groups	1	1.09	1.09	0.47	0.49	3.89
Wtn. Groups	204	473.13	2.32			
Total	205	474.22				

Interestingly, the plots show that "standing" users are slightly more inclined in collaborating with the robot, even though the statistical analysis (Table 6.3) confirms that the CA values are firmly stable, when different activities are compared. Thus, the Activity performed by the human does not appear to be a perturbing factor. The populations for the two values of the activity factor are balanced as 103 users participated for each of the configuration.

Right block of Table 6.3 reports the ANOVA results by highlighting the *p-value*, the *ratio of the two mean squares values* (F) and the *critical value* (F crit), the *sum of squares* (SS), the *degrees of freedom* (df) and the *mean square* (MS) for the given experiment. Instead, Table 6.3 reports on the left for each of the considered settings, the number of users, the sum of CA values, its average and variance.

6.3. Concluding Remarks

The aim of this chapter is to study how to include in SK4R$_E$ characteristics of the environment which are not directly observable through robot sensors. More precisely,

2. *Can we use* SK4R$_E$ *to characterize social configurations of E? Robots operating in human-populated environments have to feature the ability to represent features not directly observable through robot sensors. Hence, it is often necessary to investigate how to formalize such social configurations.*

Our response is positive and gives us the opportunity to asses that, the social configuration in which the robot is operating matters, and can significantly affect the robot chances to accomplish a task. Hence, it is important to provide the robot with the ability to recognize its social context and act accordingly.

In particular, our investigation identified which factors may influence the interaction between robot and human in the context of Symbiotic Autonomy, namely when is the robot that approaches the human to ask for help. We have characterized the Collaboration Attitude and performed two user studies to identify which factors may help the robot in improving its social skills and be included as "social landmarks"

in the robot state of the environment $s_t \in S$. These landmarks, for instance, can be maintained in a knowledge based, and can trigger particular robot behaviors anytime they are recognized in the current configuration of the environment.

Not surprisingly, and in line with the findings of several works that address proxemics as a key factor in human robot interaction, proxemics indeed plays a role. More specifically, this finding could be explained by two elements: the control that humans exercise in their Intimate space and the robot size. In fact, the presence of the robot seems to be not relevant, when the interaction takes place at longer distances. These results are particularly interesting in the framework of Symbiotic Autonomy: they suggest that a robot asking for help should approach the user in his personal space, as this distance seems to be the most comfortable for humans.

The results obtained over the Gender factor are supported by the work in [137]. In their user study, in fact, male users are more diffident and place themselves significantly further from the robot than females. These results are also confirmed in the work in [209]. Specifically, they report a considerable difference of the comfort level within the intimate area when varying the gender of the users. Their results support that males impose a dominant territory that the robot is violating, if it is positioned in their intimate areas. In Human-Human interactions, manifold psychological studies address this particular behavior. For instance, the difference in cooperating between males and females has been pointed out in [58], where this evaluation is made upon the well known *Dictator Game*. A further confirmation is provided by [203], where the gender dimension is analyzed within an experimental study of team performance. In conclusion, our results in the setting of Symbiotic Autonomy report that female experimenters show more interest in exploring a new collaboration with a robotic partner. Therefore, the robot behavior could be leveraged by allowing the robot to seek for help first by female subjects.

The height of the experimenters is another interesting feature that deserves a better investigation. In fact, few works consider the height of the robot as a *dependent variable* in their controlled studies [212, 225]. However, they do not state or highlight any empirical result on the influence of relative heights of the robot and users onto the interaction. Conversely, in our work, we noticed an interesting behavior, when classifying our users by their heights. Such a categorization has been made by considering the average among the subjects' height of our population and the $1.75m$ value has been chosen as unbiased discriminant factor. However, the outcomes of such analysis (right part of Table 6.1) could be influenced by the females which are usually shorter than males, and much more inclined to a human-robot collaboration. In fact, in our population, 70% of the female experimenters are shorter than $1.75m$, while the remaining 30% are taller than $1.75m$. Conversely, the male population is almost completely balanced. Hence, with the available data we are not able to clearly state whether the height of the experimenters plays a key role in a human-robot collaboration. In our opinion, this particular aspect deserves an additional analysis, by increasing the variability and the size of the sample, as well as the height of the robot.

While this is an interesting confirmation that space and physical approach of the robot must be carefully considered in designing also symbiotic robots, our aim was also to understand whether what the user is doing is also relevant. In other words, we intended to test whether it is worth trying to characterize the situations where it is

more effective for the robot to ask humans for help. In our first study, we studied the Context (Working vs. Relaxing) and our initial hypothesis relied upon the intuition that humans in a relaxing context are more inclined to a collaborative behavior. The results of the experiment show that there are not statistically significant differences when changing the operational context. This finding could be explained by a strong focus on the social interaction with other humans in a relaxing domain and it may suggest that robots are not yet considered "social" partners. This factor trades off the nature of the working context, where people are usually busy in their tasks.

As our first user study did not provide a clear answer to our question, we implemented a second user study trying to provide a better characterization of the situation in terms of the activity performed by the human (Standing vs Sitting). However, somewhat unexpectedly, the analysis of the data collected in the experiment indicates that humans do not show a different attitude depending upon the activity variable. As a consequence, it seems that the analysis of the situation where the symbiotic interaction takes place, does not have a significant impact on the design of robots' behavior when asking humans for help. It is worth remarking that "activities" have been determined in accordance with elements that are recognizable through robot perception capabilities. In other words, we identified activities that the robot will be able to detect through its own sensors. Such an outcome can have several justifications that we address in the following. First, the embodiment of the robot prevents the establishment of an interaction between the robot and the user at the emphatic level. Second, as a direct consequence of the previous remark, robots are not yet considered as social partners and their presence within the environment is still seen as a novelty factor. Finally, humans may consider the robot requests not plausible as they were expecting to act as operators commanding the robot. In fact, the way humans interact is strongly related to how the social partner is considered and perceived. This seems to strongly bias interactions and collaborations [47] and surely needs to be the subject of further investigation. However, by looking at our findings, we still consider that Collaboration Attitude needs to be better evaluated, including additional elements that will become available to the robot as its perception capabilities improve. Discovering new enabling factors for CA will help increasing robot's chances to be considered a social partner, when shaping social behaviors in everyday scenarios spanning from guidance in museums to assistance in shopping malls. In fact, as soon as robots will operate more frequently in human populated environments, symbiotic autonomy will play a key role in achieving a productive coexistence and thus, collaboration will become essential.

Nevertheless, we believe that a vast plethora of other factors are expected to trigger an impact on the Collaboration Attitude of the robot and influence the robot perception of social environmental features. This is an important concept and it is mandatory to be included in the robot state of the world in order to support its action planning and execution. This is key, not only from an efficiency and safety perspective, but also, to show socially acceptable behaviors in different working applications.

Chapter 7

SK4R for Multi-Robot Search

Robots exploit domain-specific representation to support action execution and long-term behaviors. Especially in multi-robot domains, such representations are fundamental as they implicitly encode a communication protocol among teammates. This chapter provides an implementation of SK4R to address multi-robot cooperation. To this end, we focus on both the implementation of *environmental module* SK4R_E, and the κ*-function parameter module* SK4R_P. In particular, the former is implemented as a distributed world model, while the latter as a set of parameters of the coordination system "designed" by an expert. We evaluate SK4R by addressing the problem of coordination within a team of cooperative autonomous robots that need to accomplish a common goal. We show how the state of the team can be distributively reconstructed, and how the parameters of the coordination system can be designed to generate collective and individual behaviors. Section 7.1 introduces and describes the proposed coordination approach based on the skar representation. Section 7.2 formalizes the used methodology and describes the implementation of SK4R in two application scenarios (Section 7.3) by reporting the obtained results in RoboCup soccer competitions [169] and in a indoor office application [170]. Finally, Section 7.4 summarizes the proposed approach and its contributions.

7.1. Multi-Robot Search for a Moving Target

Multi-Robot System (MRS) have been deeply studied during the past few years to develop effective solutions for multi-robot task execution, distributed world representation, and robust coordination. Such techniques for Multi-Robot Systems have shown to successfully handle several requirements in different environmental setups. Moreover, proposed approaches have been evaluated against varying conditions (e.g. communication bandwidth) and shown to be scalable and possibly adaptive. However, there are specific configurations of the environment ($\bar{S} \subseteq S$, as defined in Definition 4.1) in which the team strategy needs to be to dramatically adapted to the current world state s_t. In this chapter, we address the problem of multi-robot coordination by proposing a *distributed* domain-specific implementation of SK4R. We realize the environmental module (SK4R_E) by designing a distributed world modeling (DWM) approach updated through exchanged information among the

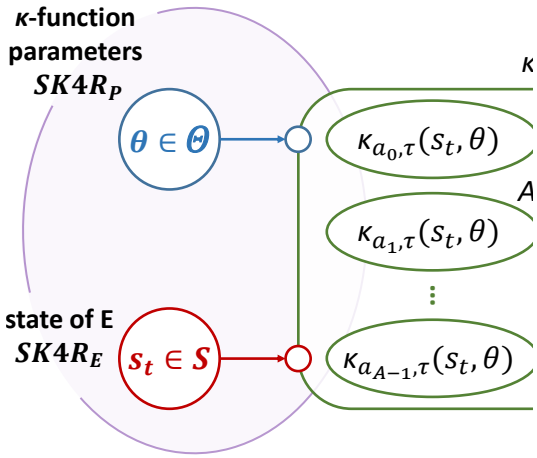


Figure 7.1. SK4R components formalized and implemented to address Multi-Robot search for a non-adversarial target.

teammates. Then, through the κ-function parameter module (SK4R$_P$), we expose some of the parameters of the coordination system that are designed to influence the distributed task assignment (DTA). Figure 7.1 recalls the overall SK4R scheme and highlights (in purple) the two modules of SK4R which are active subject of study of this chapter.

Specifically, we propose a novel approach to distributed multi-robot search which (1) reconstructs the state of the environment s_t by relying upon individual robot perceptions, and (2) influences the team task assignment process by means of algorithmic parameters. To this end, we enable the team to recognize *contexts* of operation used to select particular values of the κ-function parameters (κ-parameters for short) and influence the team actions depending on the current task τ. Our analysis of the literature shows that the proposed approaches to coordination are developed along two main directions [62, 205]. On the one hand, there is a significant body of work that can be characterized as *distributed world modeling*, where the aim is to share information so that each robot can make decisions on which action to take, based on a world model that is built through the exchange of the local views of each robot in the team [90, 235, 187]. On the other hand, methods such as *distributed task assignment* exchange information (i.e. utilities [129]) to allow each robot to choose the task that is most appropriate, considering the preferences of the teammates [144, 128, 31].

While there are sometimes applications where sharing the world model is not advisable, either because of communication constraints or because of implementation requirements [87], a suitable approach to distributed world modeling can substantially improve the coordination via task assignment. Hence, we design SK4R$_E$ as a distributed world model, that is specific to the chosen application domain, and is updated based on the information received by the teammates. In order to avoid heavy requirements on the network, our SK4R$_E$ relies upon (1) an domain-specific representation of the environment, and (2) a limited exchange of information. Taking into account the distributed world model, the system assigns the set of active tasks and coordinates the robots via distributed task assignment by highlighting areas of

Figure 7.2. Picture of the Turtlebot, Erratic and Nao robots coordinated with the proposed approach.

the environment where a particular collective (and individual) behavior supports the accomplishment of a collective task goal.

To handle the changes in the environment that might require a complete change in the strategy, we categorize any identifiable configuration of the environment as *contexts*. More specifically, we allow robots to detect *events* which are used to determine *contexts*. Such events can be received from the network, or perceived by the robots. Then, we exploit the high level representation of their distributed world model to encode contextual information, and adapt the team strategy dynamically by means of the κ-parameters. We represent contexts as a combination of two components, namely *environmental* and *task-related* knowledge [219]. Contexts can be triggered by specific events related to the operational environment and support the selection of the best team strategy. A distinguishing feature of the proposed approach, is the explicit modeling of contexts, which allows for a context switching depending on the environment monitoring (e.g. network bandwidth)

The result of this implementation of SK4R is a coordination algorithm that contributes in (1) combining a dynamic distributed world model with a distributed market-based techniques for role assignment, (2) integrating Contextual Knowledge in a multi-robot system to improve the overall performance of the team. We deploy the coordination system in two different case studies for locating a moving, non-adversarial target: we consider the problem of locating an unseen ball in a RoboCup soccer game [169] and the problem of finding a moving person in an office environment [170]. The system has been deployed on several simulated and real robots, including a team of Turtlebot and Erratic robots and a team of humanoid NAOs. Figure 7.2 shows a picture of the robots coordinated through our proposed technique. We carried out a substantial set of experiments in a simulated environment that show the effectiveness of our approach.

7.2. Distributed Implementation of SK4R

In this section, we describe our approach to multi-robot coordination combining *Distributed Task Assignment* and *Distributed World Modeling* through SK4R. The idea is to simultaneously exploit the robustness of DWM approaches and the efficiency of DTA techniques. Here, we use the former to implement SK4R$_E$ and to distributively reconstruct the state of the environment $s_t \in S$, while the latter to coordinate robots in order to achieve a task objective. The coordination approach is then enhanced by leveraging the contextual-knowledge of the scenario. In this section, we provide a formal setting which embraces all these elements.

7.2.1. Task Assignment

The core of our coordination algorithm relies upon a *Distributed Task Assignment* (DTA), based on *utility estimations*. DTA is an instance of market-based coordination [52]: at each step, the algorithm evaluates the possibility of a given robot to perform a given task according to its *utility value*. Given a set of M tasks $T = \{\tau_1, \tau_2, ..., \tau_m\}$ for a team of N robots $R = \{r_1, r2, ..., r_n\}$, a *utility estimation vector* (UEV) can be seen as the estimation of "how good" a particular robot is for each task τ_i at a given time t. If one denotes, with $b_{i,j}(t)$, the estimation that the robot r_i computes for the task τ_j at time t, its UEV can be expressed as:

$$UEV_i(t) = \Big[b_{(i,1)}(t), \quad ... \quad , \; b_{(i,m)}(t)\Big] \;\; with \; b_{(1,j)} \in \mathbb{R} \tag{7.1}$$

The *utility estimation matrix* (UEM) represents the utilities of all the members in the team, since each row i is the UEV corresponding to each robot r_i. This matrix is computed individually by each robot and it is built by gathering the vectors sent by the teammates and transposing them:

$$UEM_i(t) = [UEV_1(t), \quad ... \quad , \; UEV_n(t)]^T . \tag{7.2}$$

Given such a representation, the task assignment can be computed by evaluating each task of the $UEM(t)$ column-wise. Thus, by considering the scores of each robot r_i, we assign to a given task τ_j the robot with the highest score $b_{(i,j)}(t)$. This mapping is performed by the *coordination mapping function* Φ_i of the robot i-*th*:

$$\Phi_i : R \rightarrow T. \tag{7.3}$$

7.2.2. Distributed SK4R$_E$

Since we want to exploit the high level formalization of the surrounding scenario modeled through domain-specific SK4R$_E$ implementation, we allow the UEM to be modified based on a distributed world model. In order to build a reliable distributed representation of the current world state s_t, we need to synchronize our robots on a common representation of the environment and keep it updated over time. Hence, we define our *distributed world model* (DWM), which represents the knowledge of the world reconstructed from a set of partial local models. We refer to the DWM_j as the distributed world model locally reconstructed by the robot j-*th*. In our case studies

a robot's local model represents the probability of finding the target in a specific section of the environment, estimated through a limited and partial knowledge of the scenario. More in detail, in this distributed setting, we refer as to the state of the environment s_t as the state reconstructed from the local perception of the j-*th* robot and information about local models received from teammates, i.e. DWM_j.

Given a set of robots $R = \{r_1, r_2, ..., r_n\}$ and their corresponding local models $LM_j(t)$ at a given time t, we are able to *reconstruct* the distributed world model for the robot j-*th* as

$$DWM_j(t) = f(LM, t) \tag{7.4}$$

where f is a specific distributed model update function and LM represents the set of all the local models $\{LM_j\}_{j=1}^{N}$. The function f needs to be specified for each case scenario, as we will see in Section 7.3. To successfully apply Eq.7.4, SK4R$_E$ has to rely on a robust algorithm for reconstructing the DWM. Hence, we adopt an *event-based system* to efficiently manage the robot internal representations of the world. This system is based on the notion of *event*. These events can be either sensed by a single robot or they can be told by an external agent to the entire team. For example, an event may be represented by a door being opened or by a person telling the team that the target was previously seen in a particular location of the environment. We use these events to change the world representation. Formally, we define *model update* a function Γ that takes in input an event $e \in Ev$ and a local model $LM_j(t)$ and outputs a modified local representation of the world. Thus, the local model of each robot will be modified in the following way:

$$\overline{LM}_j(t) = \Gamma(e(t), LM_j(t)). \tag{7.5}$$

When the model update function and the *event system* are implemented and initialized in the same way on all the robots, they can share only local events, instead of communicating their entire local model. This considerably reduces the communication overhead. Finally, the distributed state of the environment s_t – generated by this implementation of SK4R$_E$ – is the DWM_i which is defined as a collection of N local models $\overline{LM}_j(t)$ reconstructed in accordance with Eq. 7.5. The algorithmic formalization of the distributed model update and dynamic task assignment is shown at the end of the next section.

7.2.3. Implementation of SK4R$_P$ – Context System

We provide an implementation of the SK4R κ-function parameter module (SK4R$_P$) as a contextual system. Accordingly, we show how contextual knowledge can increase the robot performance in the tasks to be accomplished. In fact, by detecting the current configuration of the environment, we can dynamically change the team strategy. In our approach, we use contexts to allow the robots to represent situations that require a different coordination strategy The *Context System* gathers contextual knowledge and outputs the best strategy to adopt during the execution of the team mission. Such strategy carries out high-level information directly represented as κ-parameters Θ of the underlying coordination system. These parameters are used to select the proper strategy. Formally, we characterize SK4R$_P$ as a function CS

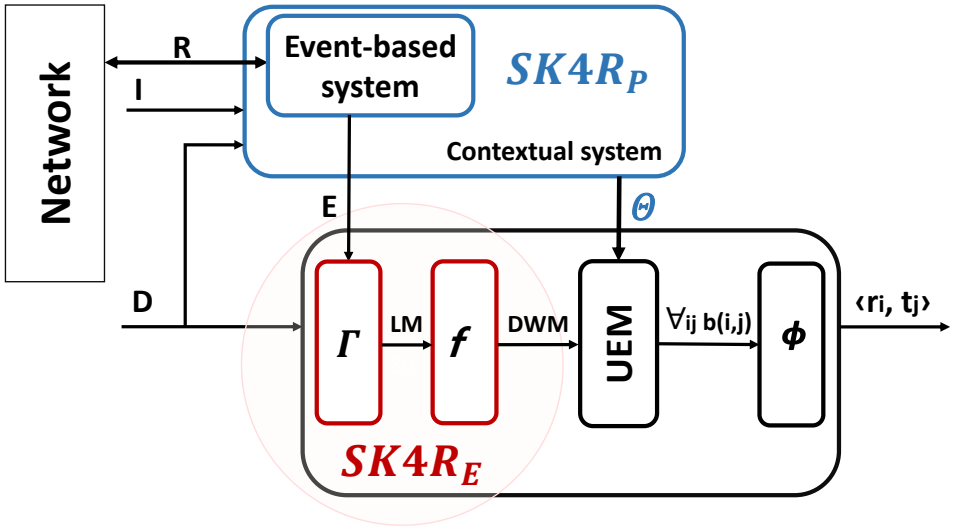


Figure 7.3. Sketch of our coordination system. The contextual system informs the team about the current context formalizing the most suitable strategy. The coordination system coordinates the robots based on the contextual information: it first updates the local models through Γ, then reconstructs the distributed world model through f. Finally, it computes the UEM outputting a mapping from robots to tasks.

which takes as input the set of sensory data D and external input events I, and, generates a coordination strategy Θ for each different context:

$$CS: [\ D\ \times\ I\ ] \rightarrow \Theta \tag{7.6}$$

The information flow of our framework is shown in Figure 7.3. The context system influences the regular execution of the coordination system by means of different *strategies* encoded in Θ parameters. More formally, at any given time t, the coordination strategy influences the mapping function expressed by Eq. 7.3 by modifying the structure of the utility matrix and/or the utility estimations of the $UEM_i(t)$ (Eq. 7.2). We refer as $\overline{UEM_i}$ to the utility matrix which has been dynamically adapted to through Θ. As shown in Section 7.3, the adaptation of the utility function is extremely effective, when the requirements of operational scenario are substantially changing.

The overall coordination protocol locally run by the single robot is reported in Algorithm 1. Specifically, the robot i-*th* detects the current context by evaluating data sources and external input events (line 1). Then, it retrieves the information needed from the other teammates (line 2) to locally reconstruct their local models (line 3). The robot is now able to estimate a global model DWM_i by considering the reconstructed local models $\overline{LM_j}$ of each connected teammate (line 4). At this point, the robot can compute the set of reference tasks according to the current state of the world (line 5) and its utility vector UEV_i (line 6). The robot multicasts its utility vector, and waits for the estimations of the other teammates (line 8) Finally, by means of Eq. 7.2, the robot calculates the utility matrix $\overline{UEM_i}$ (line 9); Then, the procedure locally computes and returns the most suitable task for the robot i-*th*.

Input: sensory data D, input events I, teammates R

Data: set of local models LM, reconstructed distributed world model DWM, context system CS, teammate j state TS_j, κ-parameters Θ, set of Task to assign T, robot i utility estimation vector UEV_i, teammate j utility estimation vector UEV_j, utility estimation matrix $\overline{UEM_i}$

Output: Task for the robot i-*th* T_i

begin

1 $\Theta \leftarrow$ updateContext(CS, I, D);
// For each teammate j receive its updated state
2 $\{ TS_j \}_{j=1}^{N} \leftarrow$ *getTeammateUpdate*(R);
3 $\{\overline{LM}_j\}_{j=1}^{N} \leftarrow$ updateLM($\{ TS_j \}_{j=1}^{N}$);
4 $DWM_i \leftarrow$ reconstructDWM($\{\overline{LM}_j\}_{j=1}^{N}$);
5 T $\leftarrow$ computeTasks(DWM_i);
6 $UEV_i \leftarrow$ computeUEV(T, Θ);
// send the utility vector to teammates
7 sendUEV(UEV_i)
// For each teammate j receive utility estimation by other teammates
8 $\{ UEV_j \}_{j=1}^{N} \leftarrow$ *getUtilityVectors*(R);
9 $\overline{UEM_i} \leftarrow$ computeUEM($\{ UEV_j \}_{j=1}^{N}$);
return $T_i \leftarrow$ mapping($\overline{UEM_i}$);

end

Algorithm 1: Context-Coordination

7.3. Application Scenarios

To prove the effectiveness of our approach, we address two settings: a soccer game during which the robots need to search for a moving ball and coordinate in order to score against an opposite team (Section 7.3.1), and an office environment, where a team needs to locate a person in a non-adversarial setting (Section 7.3.2).

7.3.1. RoboCup Soccer Competitions

Our approach to coordination has been first deployed on a team of NAOs. NAOs are commercial, autonomous, 25-DOF humanoid robots. They are equipped with a wide variety of sensors and actuators, including two CMOS cameras, multiple proximity sensors, four microphones, and two speakers. In this setting, a team of robots plays in a 9×6 meters soccer field of the RoboCup Standard Platform League. In our coordination algorithm this field is represented as an occupancy grid. Each cell in this grid features a score, representing how likely it is to find the ball inside it. Figure 7.4 shows the DWM reconstructed by a team of NAOs in a simulated environment.

In this setting, we arrange possible configurations of the state space (i.e. working contexts of the system) on a tree structure composed by two main branches, namely the *task-related* and the *environmental* branches, shown in Figure 7.5. Each of the branch is composed by two layers determining specific configurations of the

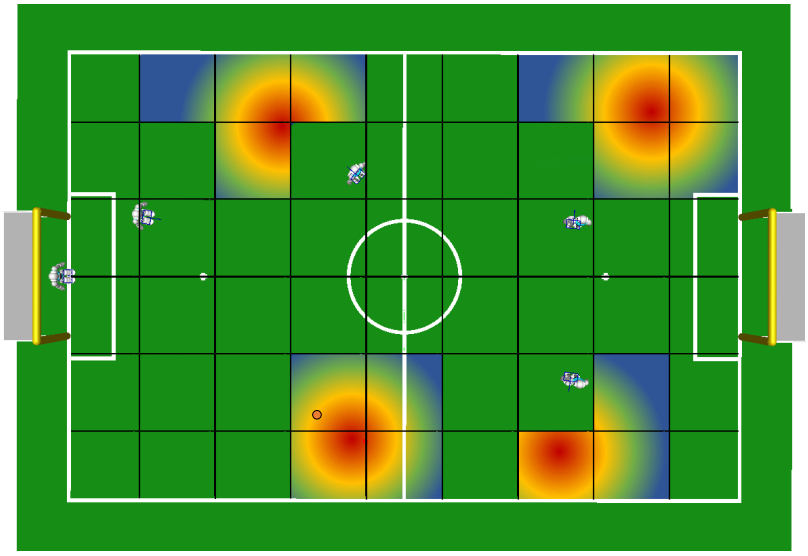


Figure 7.4. Distributed World Model for the soccer scenario. The model assigns a score to each cell encoding likelihood to contain the ball (orange in the picture).

environment. In the *task* branch, we encode a set of three basic contexts called *task-related contexts* (C_T):

- **Playing**: the robots know the current location of the ball, and the robot coordinate according to the default task-space comprising the common task in a soccer scenario, i.e. striker, defender, supporter and second supporter;
- **Ball-Lost**: the robots do not know the ball position and cooperate in order to minimize the time in locating the ball;
- **Throw-in**: the robots are searching for the ball in which the ball has rolled out of the field. Hence, they can modify the search strategy by exploiting particular rules governing the soccer scenario. In fact, such a context is triggered by an external Game Controller [1]

In the environmental branch, we instead characterize the world depending on the network reliability that allows us to define another two contexts, called *environmental contexts* C_E:

- **Network up**: the robots are in a suitable network condition, i.e. the messages exchanged among the robots are received in a fixed amount of time;
- **Network delayed**: the current network condition does not allow a reliable communication among the robots.

When a context is recognized the robots start coordinating and sharing information. In the soccer scenario, the robots share the outcome of two actions as events, namely *clear area* and *ball found*, which are associated with the centroid of a visited

[1]The Game controller is an external electronic referee used to communicate with the playing robots during a regular match.

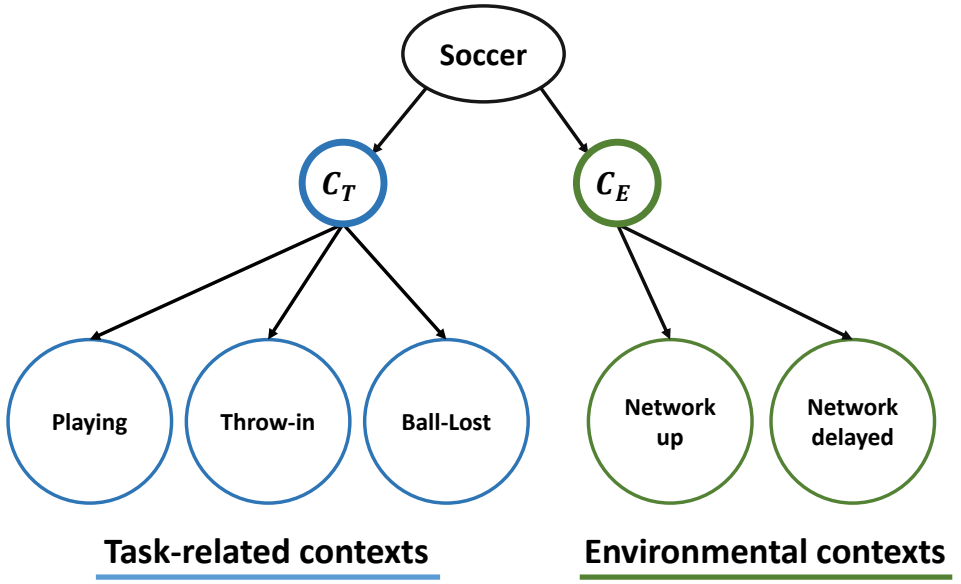


Figure 7.5. Context hierarchy to model configurations of the environment for a team of humanoid NAO robots during RoboCup soccer competitions.

areas. Specifically, such events are locally detected by each robot. Upon detection the agent sends a message to the teammates, which when received it is used to update the distributed model of the robots according to the event type (Eq. 7.5). For instance, a "ball clear" event has the effect of reducing the probability of finding the ball in a given area. In particular, while searching for the ball, the robots exchange the centroids of controlled areas. When the robot i-*th* receives an event messages by the robot j-*th* and updates its model, then it merges the new information in the global distributed model DWM. To this end, Eq. 7.7 shows how the reconstruction function f (Eq. 7.4) is implemented in this case. This function is defined as the union of the score of each cell, updated as:

$$DWM_i \;:\; \forall x, y \; cell_i^{\langle x,y \rangle} = \underset{cell_j \in LM_j}{\arg\min} \{score(cell_j^{\langle x,y \rangle})\} \tag{7.7}$$

where the score of the $\langle x, y \rangle$ cell in the overall representation of the robot i-*th* is the minimum score among all the local models LM_j of each robot. Intuitively, it informs the i-*th* that one of its teammate has recently controlled a given area and the search can be directed elsewhere.

When the events are received from the team, the Context System determines the current context outputting a set of parameters Θ used to leverage the team strategy. In this specific scenario, κ-parameters are used to activate the set of tasks that are related to the detected context in accordance with Eq. 7.2. We define a utility function that given a robot and a task, it returns a score $b_{(i,j)}$ in Eq. 7.1 representing how suitable the robot i-*th* is for the j-*th* task. Depending on the current configuration of the environment, the utility function exploits different κ-parameters $\boldsymbol{\theta} \in \Theta$ *designed* before robots deployment. It yields the utility function implemented as

$$
\begin{aligned}
\mathbf{b}_{(i,j)} = \; & \theta_0 \cdot \textbf{distance-to-ball} \\
& + \theta_1 \cdot \textbf{distance-to-task}_\textbf{j}\textbf{-position} \\
& + \theta_2 \cdot \textbf{orientation-to-task-position} \\
& + \theta_3 \cdot \textbf{time-since-role-taken} \\
& + \theta_4 \cdot \textbf{role-bias} \\
& + \theta_5 \cdot \textbf{battery-level} \\
& + \theta_6 \cdot \textbf{distance-to-search-area1} \\
& + \theta_7 \cdot \textbf{distance-to-search-area2} \\
& + \theta_8 \cdot \textbf{distance-to-search-area3} \\
& + \theta_9 \cdot \textbf{distance-to-search-area4}
\end{aligned}
\tag{7.8}
$$

where each κ-parameter θ_p is chosen empirically and regulates the different components of the utility function. In this case, Eq. 7.8 takes into account the euclidean distance between the ball and the robot; the distance between the robot and a target position; the elapsed time since the current role was assigned; and a bias term for each robot (such bias is used to solve ambiguous situations); the robot battery level; and the distance to the centroid of search areas generated by Eq. 7.7. It is important to remark, that $\boldsymbol{\theta}$ are adapted upon contexts detection, and leverage the robots behavior to fulfill the requirements of the current team task τ. For example, let us consider the setting in which the robots are searching for the ball. In the "Search" configuration, while SK4R$_E$ reconstructs the DWM for each robot, the task assignment routine is adapted by means of the SK4R$_P$ output, i.e. the $\boldsymbol{\theta}$ parameters. Such parameters are used to weight components of the utility function that are significant to address the current working context. Operationally, to adapt the system to the "Search" configuration, we set $\boldsymbol{\theta}$ to be

$$\boldsymbol{\theta} = \{0.0,\ 0.0,\ 0.0,\ 0.05,\ 0.1,\ 0.05,\ 0.2,\ 0.2,\ 0.2,\ 0.2\}.$$

Intuitively, in the "Search" context, both distances to the ball and default playing positions, are not relevant to the task and can be discarded when computing the utility values. It is important to remark that, while Θ are retrieved by SK4R$_P$, the world model is instead reconstructed dynamically from individual robot perceptions.

Network Monitoring. In real applications, the robot communication is one of the main problems and it is not typically monitored at any time to adapt the coordination. In RoboCup, network reliability is one of the main issues to be tackled in order to design a workable coordination framework. In this perspective, multiple proposed approaches evaluate their performance against unstable network conditions and limited band-width. Several solutions adjust the team displacement in order to maintain connectivity [166] or periodically share information to bound the network overload [85]. However, to the best of our knowledge, there is no approach that continuously performs an on-line analysis of the network bandwidth to select the most appropriate coordination strategy according to the current network reliability.

The network condition is detected directly by the team of robots by evaluating the *Round Trip Time (RTT)* of the packages shared among the team. Such evaluation is useful to detect the current network context that can assume three different values, namely, *unreliable network* and *reliable network*. Hence, the recognized context is used to select the best configuration of the Θ parameters. In this case, the parameters influence the system on two levels. First, one of the parameters is used to regulates rate of sending network packages to other teammates. Such a parameters is set to 10 *packages/second* when the network is reliable, and 1 *packages/second* otherwise. Second, the κ-parameters influence the utility estimation by assigning more value to the *role-bias* term weighted by θ_4 in Eq. 7.8. In critical network conditions, such a bias has the effect to decrease reactiveness of the system, but provides robustness and stability in selecting robot roles – which is critical in hostile network settings. Next section will show the effectiveness of adapting the κ-parameters to the current working context.

Experimental Evaluation

The virtual environment where experiments are carried out is part of the *B-Human architecture*, which provides a RoboCup-dedicated simulation platform entirely written in C++[2], that features a rather accurate model of the behavior and capabilities of the humanoid robot in the field. In the soccer case study, our goal is to show the effectiveness of the system in preserving team performance despite adversarial working context. For example, when none of the teammates perceives the ball in this highly changing environment. Accordingly, we first compared our context-aware coordination against a team that cannot distinguish between a "Throw-in" and a "Ball Lost" context. Both teams implement the same distributed world model (DWM), and share the same combined coordination model; however, the red team is not equipped with a contextual system. We measured the cumulative time during which the ball was not seen by the team in a game (i.e., 10 minutes). Figure 7.6 reports the results averaged over 100 runs for the two different contexts considered.

Our algorithm was able to recognize the contexts and specialize the search, thus resulting in an overall better performance. It is worth noticing the effects of the different levels of information available in the two contexts. In fact, when more detailed information was provided (e.g., in the "Throw-In" context), an increase in performance is noticeable. Instead, in both contexts the red team was not able to exploit the available information, always performing an *uninformed search* in all of the different contexts.

In a second experiment we measured different strategies while varying the reliability of network communication. More specifically, we developed an external tool for artificially introducing network delays in the simulations. We allow our team to detect the *unreliable network context* when the delays were above a certain thresholds. As in the previous test, in this setting the red team implements the same underlying coordination system, but, it is not able to detect network contexts. In this scenario, we adjust the strategy of our team to have a more aggressive formation, send a restricted number of packages to reduce the network overload of the team, and assume more static behaviors in order to reduce errors in the transmission of

[2]https://www.b-human.de

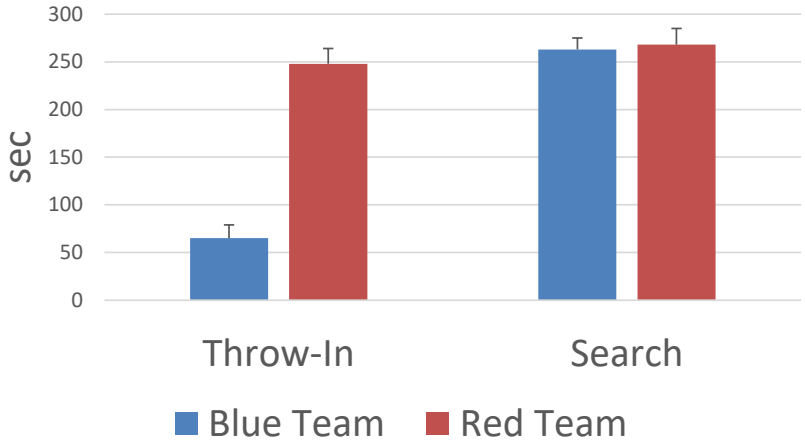


Figure 7.6. Cumulative time during which the ball was not seen in a 10 minutes game for the two contexts "Throw-In" and "Ball-Lost". The results were averaged over 100 runs.

processed data. As we do not gave a specific *task to test*, we can only verify the quality of the role assignment by considering the scores of the games. Table 7.1 reports the results obtained in 173 runs of the experiments.

Table 7.1. Game results of the blue team over 173 runs of a soccer match (i.e. 10 minutes).

	wins	losts	ties	games
blue	95	36	42	173

The results show a considerable difference in the number of won matches for the blue team that were able to categorize network contexts and adjust its coordination strategy dynamically. The scores prove that an adaptation on the operational scenario is always preferable when possible. Our coordination algorithm has been firstly validated on extensive testing sessions, and then implemented on real NAOs to allow the team of robots to compete during RoboCup matches. To the best of our knowledge, this is the first example of coordination that is adapted by a continuous monitoring of the network performance.

7.3.2. Indoor Office Scenario

In this subsection, we instantiate our framework to the problem of multi-robot target localization by applying our system to different operational scenarios. Several works address the problem of pursuit evasion [2]. Here, we focus on non-adversarial target localization [86, 72]. However, proposed solutions do not formalize a distributed world model which, is given to the team before operation and, remains known and static. Most importantly, none of the existing approaches adapt the searching strategy depending on the current world state. Hence, we consider a complex setting, where the robots in the team coordinate to find a person in a given scenario. As shown in Figure 7.7, we discretize the office environment through a *topological graph*, thus showing the effectiveness of the approach in a completely different representation

Figure 7.7. Distributed World Model for the indoor office scenario.

of the environment.

The Context System is implemented as a search on a decision tree and a knowledge base is used to recognize the set of contexts $\{Meeting, Lunch, Morning, Afternoon\}$. This knowledge base includes information about the scenario, such as scheduled meetings, habits, or room and object positions. We exploit contexts to assign different weights to promising nodes to look at, and influence the search accordingly. To this end, we pair to each node of the graph a θ parameter that modifies the likelihood of node to contain the search target, and thus, can guide the search strategy of the team. Additionally, we semantically label the environment where the robots operate. In this way, we are able to perform spatial reasoning about objects and rooms, which helps carrying out the task.

In this case, even though the set of *task-related context* are determined by the daytime or daily meetings, the set of action outcomes to share among the teammates is wider due to the more complex nature of the environment. In this scenario, the robots share the following events *target near location*, *door opened*, *door closed*, *clear area*, and *person found*. Specifically, *target near location* is multicasted if one of the robots is informed that the target has been seen near a particular location. Instead, *door opened* and *door closed* are communicated whether a robot perceived the status of a door has changed. Finally, *clear area* and *person found* are respectively shared when a node has been visited and when the target is found. These information are associated with a set of nodes of the topological graph. Thus, the team can reconstruct the DWM by applying:

$$DWM_i \;:\; \forall n \; node_i^n = \underset{node_j \in LM_j}{\arg\min} \{score(node_j^n)\} \tag{7.9}$$

which states that for each node n in the distribute world model of the i-*th*, the associated score is the minimum found in all the local models.

According to Eq. 7.2, we coordinate the robots based on their utility values, with respect to a given set of *most-likely* nodes. The utility score of each pair $\langle r_i, \tau_j \rangle$ (i.e.

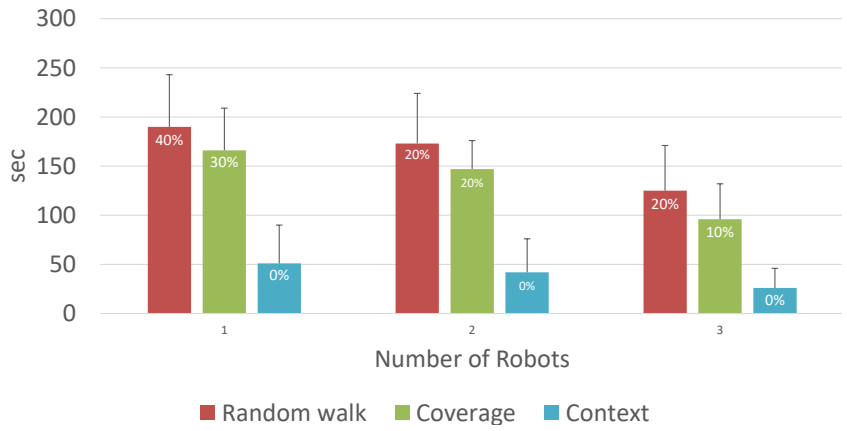


Figure 7.8. Average time needed to locate the target for the considered algorithms. The percentages represent the ratio of failed tasks.

$\langle robot_i, node_j \rangle$) is computed according to the cost of the path connecting the robot *i-th* and the node *j-th*. We use the Dijkstra algorithm for searching the optimal path p^* between two given nodes and to evaluate the best mapping from robot to nodes. In this case, contexts are used to prioritize the areas to look at according to the scheduling of the searched subject. For instance, if the team is within a *meeting context*, then the nodes that are associated to the person's office are reranked as the most promising areas to look at.

The office experiments are carried out in the STAGE simulator by implementing our coordination system within the ROS framework. In this case, a team of mobile bases had to search for a person in the office environment. In this setting, we varied the number of robots performing the search, comparing our approach with other algorithms for exploration. Specifically, we compared it with two search strategies: a *random walk*, where the robots randomly explore the environment without coordinating; and a *coverage* search, where the team uses a DTA to coordinate, by keeping track of the visited nodes and randomly choosing the next ones to be explored. Figure 7.8 illustrates the average time in seconds needed to find the moving target. This measure has been averaged over 10 runs for each configuration. The experiment was recorded as a failure, if the team needed more than $300s$ to complete the search.

The results of the experiments reported in Figure 7.8 show that with the proposed approach the performance improves as the number of robots grows and the information shared increases. Indeed, contextual information helps to properly evaluate the dynamics of the environment – through κ-parameters – and to rerank areas to look at in accordance with the current context. The results confirm our hypothesis, as the average time in locating the target considerably decreases, when different search strategy can be properly exploited. Overall, our approach performed better in all of the considered configurations in terms of both time needed to complete the algorithm and percentage of successful tasks.

7.4. Concluding Remarks

In this chapter, we considered the problem of coordinating a team of autonomous robots capable of identifying configurations of the environment (i.e. contexts) they operate in. The work proposed helped us in addressing one of our research question listed in Section 4.3 in Chapter 4:

3. *The robot behavior is actively adapted to the state of E, however, is it possible that the robot behavior influences how the state of the environment is generated? How can such a mutual dependency be implemented in a robotic system? Is it possible, for the robotic system, to feature an active interaction of the* $\mathrm{SK4R}_E$ *and* $\mathrm{SK4R}_P$ *components?*

To this end, we contributed an approach that allows for a distributed modeling of the environment – as domain-specific representation of $\mathrm{SK4R}_E$ for the multi-robot scenario. Such a model is then used to adapt the coordination system to dynamic changes of the scenario by means of the κ-function parameters. Accordingly, we exploit contextual knowledge to categorize the state of the environment and improve the team effectiveness of the team by exploiting different configuration of the κ-parameters. The contents of this chapter confirm that the behavior of the robot can influence what may (or may not) be included in the current state representation of the team. This is a crucial point for κ-agents and suggests that interactions between $\mathrm{SK4R}_E$ and $\mathrm{SK4R}_P$ have to be carefully engineered. For example, in our scenario, different strategies, triggered by the contextual system, influence how the team explores the environment, and thus, actively change how they reconstruct their distributed world models.

To demonstrate the benefits of such an active interaction between the two modules of SK4R, our coordination algorithm has been applied to the problem of locating a moving, non-adversarial target in two different settings. We successfully deployed our coordination system on multiple robots: specifically, our experiments report the performance of our contribution on a team of NAO robots in a soccer scenario and on a team of mobile bases in an office environment. In both scenarios, we found a significant reduction in the time needed to find the target, underlining the effectiveness of the approach. More specifically, as opposed to previous work aiming at developing methods that can scale up with respect to varying factors (e.g. communication bandwidth, delays), we propose an approach where the system can handle the changes in the operational scenario and select the best strategy online.

We have focused on the formalization and implementation of two modules of the SK4R representation. Namely, a distributed world model and the contextual system used to generate different strategies of the team encoded as κ-parameters. As a future work in this field, we are investigating the problem of representing contexts that need multiple and non-deterministic perceptions to be recognized, and to allow the team of robots to handle situations where the construction of the world model became more challenging. In fact, in the proposed scenarios, the coordination system assumes both contexts and events as pre-defined by an expert user. However, in unknown and unstructured environments they cannot always foreseen. To overcome this issue we want to investigate methodologies to discover contexts and events

during robot mission by adapting the team strategies to the current scenario and robot mission.

Chapter 8

SK4R for Optimistic Planning

A domain-specific representation is key to enable effective robot behaviors and succeed in different applications. However, in order to simplify the planning process, robot actions are often considered as atomic actions, and their *affordance semantics* is usually pre-defined by experts of the domain of application. Hence, to generalize to more complex and dynamic applications, we investigate the problem of enabling a robot to understand the effects of its actions on the environment. In this chapter, we focus on the SK4R$_P$ module and investigate the problem of incrementally update the κ-parameters (i.e. $\boldsymbol{\theta}$) in order to learn affordance semantics. More in detail, Section 8.1 contextualize the contribution of this chapter within the current state-of-the-art. Section 8.2 introduces LoOP, our iterative learning algorithm for optimistic planning. We describe its building blocks and present its components in detail. Section 8.3 demonstrates the robustness and flexibility of LoOP in different robotic domains and presents an extensive experimental evaluation. Finally, Section 8.4 concludes the chapter by discussing contributions of LoOP and future directions.

8.1. Robot Policy Learning

Robots have to show robust behaviors to complete cognitive tasks in different scenarios, such as service robotics and uncontrolled industrial environments. However, deciding the best action to perform, is a complex task due to unpredictabilities of the physical world, uncertainties in the observations, continuous state spaces and rapid explosions of the state dimensionality. Moreover, the task objectives, the structure of the environment and the robot embodiment can heavily affect both the applicability of decision-making techniques and their overall performance. In this setting, reinforcement learning (RL) approaches have successfully applied to enable robots to act and improve their performance over-time. Hence, in this chapter, we address the problem of learning effective robot behaviors by exposing specific parameters of the learning process, which are used to learn and encode the affordance semantics of robot actions. We build upon the SK4R$_E$ representation and dedicate this chapter to the study of the κ-functions parameters module – SK4R$_P$. To this end, we assume state of the environment s_t generated by SK4R$_E$, and tackle the problem of learning κ-parameters (i.e. $\boldsymbol{\theta}$) through direct robot experience, in the

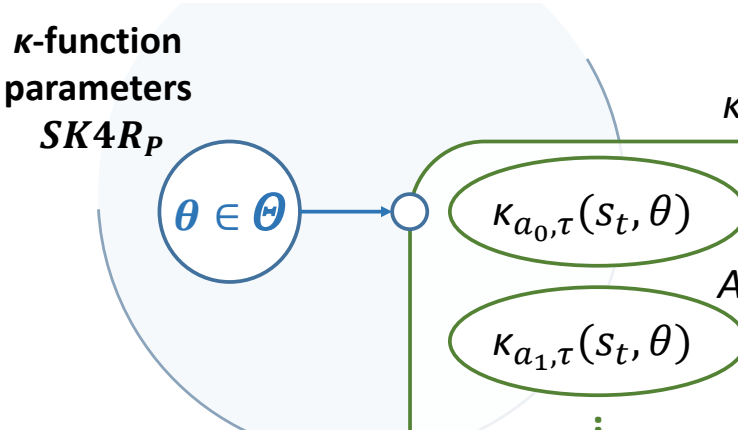


Figure 8.1. SK4R κ-function parameters module – SK4R$_P$.

context of reinforcement learning. Figure 8.1 recalls the overall SK4R representation and highlights (in blue) the subject of study of this chapter: SK4R$_P$.

The applicability of RL methods in robotics, however, is not straightforward and may suffer from several factors related to partial observability and high-dimensionality of the state-space. In fact, in order to handle the "curse-of-dimensionality" and to guarantee effective behaviors in complex scenarios (e.g. hyper-redundant robot, multi-robot settings), we need to face several issues: (1) state generalization, (2) exploitation of prior knowledge and (3) exploration of the state-space.

To solve generalization and policy generation, several methods take advantages from recent success of deep neural networks (DNNs) (or in general by employing function approximators) both in RL [134, 121, 135] and robot planning [120, 119]. Alternatively, other approaches run different simulation in parallel to collect multiple robot expirience in a reasonable amount of time [189]. Still, these techniques require the agent to explore its vast state-space, and need a remarkable amount of "good" training samples to synthesize competitive action policies. Moreover, it is difficult to encode prior knowledge through function approximators. This, instead, is usually achieved with learning from demonstration techniques [29, 223, 151], but too often with little performance guarantees when demonstrations are not properly transferred in the robot action-space. To address this issue, in fact, planning techniques such as Monte-Carlo Tree Search (MCTS) have bee used [26]. However, these approaches suffer the generalization problem not being effective in relating similar states [198].

Most importantly, in robotics, the lack of a big number of training sample, and the sparsity of good ones, are limiting conditions that do not allow state-of-the-art approaches to be also effective and practical. In fact in robotics applications, generating competitive policies with few training samples is an extremely appealing challenge. In this chapter, we contribute a solution to address such a problem. We introduce LoOP, an iterative reinforcement learning algorithm for optimistic planning, where actions are chosen optimistically and the robot learns a policy directly from experience and interactions with the external world. To this end, we base our presentation of LoOP upon previous work [175, 177, 35, 172, 173]. We attack the generalization problem in policy generation by pairing the Upper Confidence Tree (UCT) algorithm [107] with an external action-value function

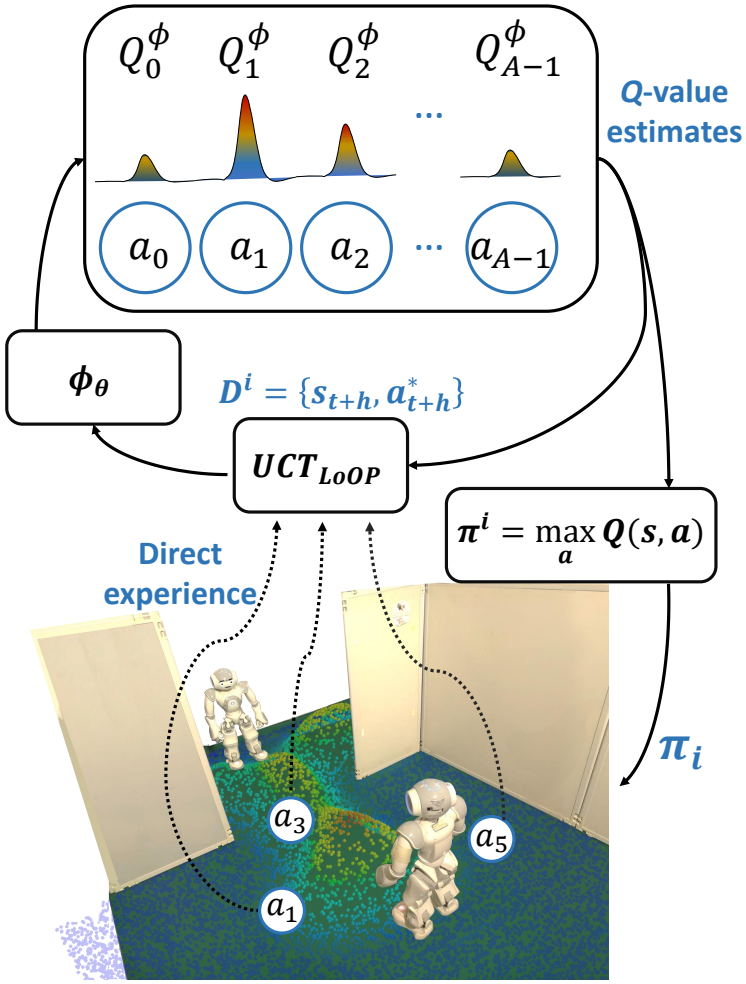


Figure 8.2. Affordance semantics learned by LoOP in a door-passing scenario with two humanoids NAO robots. The scheme illustrates how the algorithm collects direct experience with the environment, runs the Monte-Carlo search and updates Q-values estimates to generate an action policy.

approximator. Such an approximator is used in the expansion phase of the MCTS standard procedure to select "admissible" actions, and drive exploration during episode simulation. Figure 8.2 shows how LoOP (1) evaluates the current state of the world and (2) generates $\boldsymbol{\theta}$ parameters, in order to provide the robot(s) with actions that are expected to drive towards task completion. The algorithm promotes actions with a significant affordance semantics (AS) in the current state s_t by means of the function approximator ($\Phi(\cdot)$), which outputs a set of Q-values that are used for robot planning. Such a structure enables LoOP to be a general algorithm that is independent from to robot platform, the environment of deployment and function approximator used. We show its applicability in different use-cases and its flexibility with respect to two types of function approximators, namely *Gaussian Mixture Models* (GMMs) and *Deep Neural Networks (DNNs)*. While, GMMs have successfully been employed to model non-linearities in manifold applications [186], recent advancement in deep reinforcement learning [220, 200] show remarkable

improvement in generalizing unseen states and generate action policies robustly. Hence, we formalize LoOP to be modular and we leave the choice of the approximator to depend on the problem at-hand. Nevertheless, the comparison between the two approximators remarks the better generalization capabilities of DNNs, and their ability to easily address high dimensional problems. Thus, representing a better solution to the "curse-of-dimensionality".

In particular, LoOP uses UCT to explore the state space and exploits both Q-learning and a generic function approximator to model action values, which are iteratively refined by aggregating [186] samples collected at every timestep. Our goal, in this chapter, is to demonstrate that LoOP can efficiently be used to generate policies and restrict the search space in order to support action planning. We evaluate LoOP on a set of tasks involving different platforms, such as an hand-over with a fetching task with a light weight 7 DOF Kuka arm; a humanoid NAO robot; a door-passing task with two mobile robots; a navigation task; and in learning effective positioning for a defending robot in RoboCup soccer competitions. The experimental evaluation shows the effectiveness of LoOP to represent action values and boost the exploration phase. Such important features make LoOP a practical solution for robotic applications – especially when DNNs are used. Our main contributions consist in (1) the introduction and the generalization of the LoOP algorithm based on previous work, (2) and extensive experimental evaluation in both simulated and real environments.

The reminder of the chapter is as follows. Section 8.2 introduces theoretical background and describes the LoOP algorithm in detail. Then, Section 8.3 presents the experimental evaluation in simulated environments, and shows the robustness of the generated policy when transferred to the real platforms. Finally, Section 8.4 concludes the chapter with final remarks, analyzes current limitations of the approach, and points toward future directions.

8.2. LoOP: Iterative Learning for Optimistic Planning

Through LoOP, we address the problem of making planning and learning in robotics practical and usable. Our goal is to combine state-of-the-art techniques and alleviate their computational time and complexity in order to guarantee competitive action policies with few training samples.

To this end, we combine Monte Carlo Tree Search and Q-learning. Monte Carlo planning has been widely used in different simulated scenarios including complex games. For example, The UCT algorithm achieved the first success in competing in the game of Go [227]. We refer to MCTS approaches and UCT algorithm as introduced in Section 2.3.3 in Chapter 2. However, in order to generate accurate estimations, UCT needs to evaluate state-action pairs multiple times. This, unfortunately, forbids the naive use of UCT in robotic scenarios. Hence, we enable the use of UCT by exploiting learned action values directly in the expansion phase. We use such values to crop out branches of the search tree (i.e. actions) that are not expected to lead to a good ending states, which result in a remarkable reduction of the number of state-action evaluations. Then, every time UCT finishes the backpropagation phase it pairs an action to the current state which is used to

simultaneously update UCB1 values and refine Q-values for next iterations. Finally, in order to generalize to unseen states we make use of function approximators that enable the agent to take decision when the system is in a newly discovered portions of the state space.

We frame LoOP as a Markov Decision Process (MDP) (as introduced in Section 2.3.1 in Chapter 2), where decision-making is expressed as a tuple

$$MDP = \langle S, A, \mathcal{R}, \mathcal{T}, \gamma \rangle, \tag{8.1}$$

where S is the set of discrete states of the environment, A represents the set of discrete actions, $\mathcal{T} : S \times A \times S \rightarrow [0, 1]$ is a stochastic transition function that models the probabilities of transitioning from state $s \in S$ to $s' \in S$ when taking action $a \in A$, $R : S \times A \rightarrow \mathbb{R}$ is the reward function, and γ is the discount factor in $[0, 1)$. In this setting, transitions and rewards are assumed to be Markovian – i.e., a function of the current state – and decisions are represented through a policy π, that defines the behavior of an agent. Given an MDP, the goal consists in finding a policy $\pi(s)$ that maximizes its expected cumulative reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$ over a finite or infinite time horizon T. This can be obtained by using the notion of action-value function $Q^\pi(s, a)$ and by solving its corresponding Bellman optimality equations [208]

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} \mathcal{T}(s, a, s') \max_{a'} Q^*(s', a').$$

Intuitively, the action-value function represents the value of the expected return when taking action a in state s and then following policy π. It is important to highlight that, given the formalization of the action-value function $Q(\cdot, \text{cot})$, we can use Q-values estimate to encode the affordance semantics of robot actions. In fact, Q-values adhere to the definition of affordance semantics as referred to in Definition 4.1 in Chapter 4. Hence, we use Q-values to annotate portion of the state-space that support task completion, and to suggest actions to execute in a particular state of the environment s.

8.2.1. LoOP

The information flow traversing the LoOP algorithm is illustrated in Figure 8.3. It is composed by three blocks which elaborate the three main steps of the algorithm: the roll-in, the UCTLoOP search and the Q-values update through the function approximator. The first step serves to initialize the state of the system s_i and provides a set of samples from which the Monte Carlo search is started. The algorithm lets the system roll-in an ending state s_t by following a desired policy $\pi_{(roll-in)}$ that can be arbitrarily chosen. It our implementation we make use of π_{i-1} the policy generated at the previous iteration – being π_0 randomly initialized. The second step runs the UCT search enhanced by the Q-values estimate. The algorithm assumes as root of the search tree the state resulting by the roll-in phase s_t and starts the search from it. The UCT selects admissible actions and expands only those ones. Then, after the UCT backpropagation phase, the algorithm pairs to the root state the action resulting with the best UCB1 value. We will refer to this modified version

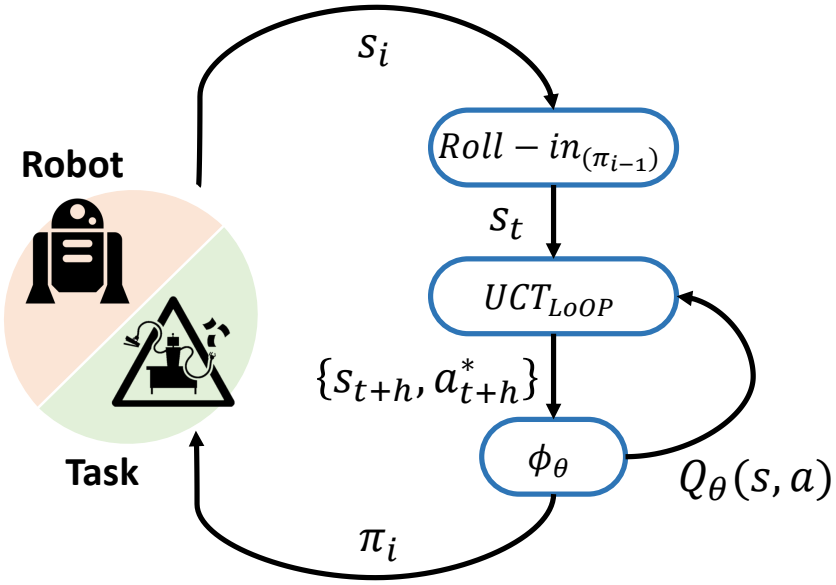


Figure 8.3. Information flow at each iteration of LoOP. The algorithm assumes an initial state of the environment s_i which depends on the robot (orange) and the task to accomplish (green). LoOP (i) evolves the system, with π_{i-1}, for $T_{roll-in}$ timesteps until state s_t, in which the UCT_{LoOP} search is ran; (ii) uses the MC search to generate a set of H transitions that associates, to each visited state s_{t+h}, the action a^*_{t+h} explored by UCT_{LoOP}; (iii) then, these transitions are (1) aggregated to the original dataset and (2) used to update the $\boldsymbol{\theta}$ parameters of $\Phi_{\boldsymbol{\theta}}$. Finally, the policy π_i, generated at iteration i, guides the robot to select an action in the environment and proceed towards task completion.

of the vanilla UCT algorithm as UCT_{LoOP}. During the last step, LoOP uses the generated pairs to update the Q-values estimate and retrain the policy. After this, a refined policy π_i is generated and exploited by the agent in the following iteration.

To accumulate new experience and make use of training samples, the algorithm performs an aggregation [186] procedure. Specifically, at each iteration, LoOP maintains a dataset $\mathcal{D}^i = \{d\}$ of transitions that are aggregate as

$$\mathcal{D}^{0:i} = \{\cup\, \mathcal{D}^j \mid j = 0 \dots i\}, \tag{8.2}$$

and used during learning. Despite the function approximator Φ employed, the dataset $\mathcal{D}^i$ is used to generate a new policy by minimizing the ℓ_2-loss:

$$\ell_2(r_{t+1} + \gamma \max_{a'} Q_\theta(s_{t+1}, a'), Q_{\boldsymbol{\theta}}(s_t, a_t)), \tag{8.3}$$

where $\boldsymbol{\theta}$ are the parameters of the generic function approximator. Thus, they constitute the main vehicle to influence the output of the function approximator, and to update the AS of an action. For example, they are represented as the mean, covariance and prior in the case of GMM, or equivalently, the network weights in the case of DNNs. in the following sections, we describe more in detail how the $\boldsymbol{\theta}$ parameters are learned – both in the case of Φ_{GMM} and Φ_{DNN}. We refer as to Φ_{GMM} as the function approximator implemented with Gaussian Mixture Models, and as to Φ_{DNN} as the one implemented with Deep Neural Networks.

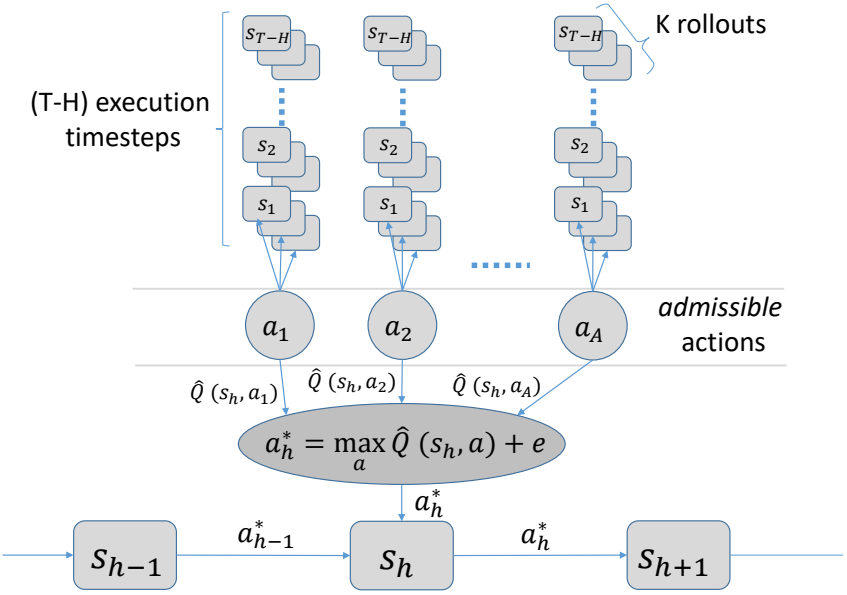


Figure 8.4. UCT execution at a state s_h generated by following the policy π_{i-1} (roll-in). The state is expanded by the algorithm for H iterations. Admissible actions are selected in accordance with their relative expected return Q. UCT, then, expands the promising actions by running a number K of different roll-outs, and selects the best action a_h^* such as $a_h^* = \max_a \hat{Q}(s_h, a) + e$.

Algorithm

A key contribution of LoOP is the combination of MC planning and Q-learning that allows to (1) improve a policy π iteratively, and (2) learn action values used to achieve focused exploration. Algorithm 2 presents the LoOP procedure in detail. The algorithm takes as input an initial policy π^0 and, for I iterations, computes the following algorithmic steps:

1. **Roll-in.** The agent evolves the policy π_{i-1} for T timesteps, which terminates in a given state $\{s_t\} \in S$;
2. **UCT_{LoOP}.** At the end of each roll-in, the agent runs a modified UCT search [107] – that we refer to as UCT_{LoOP} – for H steps, where H is the UCT search horizon. As represented in Figure 8.4 the UCT algorithm performs a set of K roll-outs for each admissible actions and returns its expected value. More in detail:
   (a) UCT_{LoOP} evaluates and generates the set $\tilde{\mathcal{A}}_s$ of admissible actions in $s_{t+(h-1)}$. Actions are evaluated according to their Q-value estimate $Q_\theta(s_{t+(h-1)}, a)$, and only actions that score a sufficient Q-value are selected to be expanded. In particular, a threshold upon the Q-values is set and all the actions that have a lower Q-value are disregarded

$$Q_\Phi(s_{t+(h-1)}, a) >= \lambda * \max_a Q_\Phi(s_{t+(h-1)}, a) + \epsilon_{\tilde{A}_s} \tag{8.4}$$

where λ is typically initialized to 0.5 and increases with the number of iterations as the Q-value estimate becomes more confident, and Q_Φ represents the Q-estimation through the chosen function approximator Φ. Nevertheless, a certain amount of exploration is guaranteed by an exploration term $\epsilon_{\tilde{A}_s}$ that accepts action with an epsilon probability;

(b) When $\tilde{A}_s$ is computed, UCT_{LoOP} expands all the actions $\tilde{a} in \tilde{A}_s$ and evolves the policy for K monte carlo roll-outs by following a ϵ-greedy policy based on π_{i-1}. When a termination state is reached, it backpropagates their expected reward in order to calculate their UCB1 value. The best action $a_h^* \in \tilde{\mathcal{A}}_s$ is then chosen according to

$$\begin{aligned} E &= c \cdot \sqrt{\frac{\log(\sum_a \eta(s_{t+(h-1)}, a))}{\eta(s_{t+(h-1)}, a)}} \\ a_h^* &= \arg\max_a Q_\theta(s_{t+(h-1)}, a) + E, \end{aligned} \tag{8.5}$$

where c is a constant that controls the exploration term E, and $\eta(s_{t+(h-1)}, a)$ is the number times that the action a is selected in $s_{t+(h-1)}$.

(c) At each $h \in H$, UCT_{LoOP} generates a state-action pair $\langle s_{t+h}, a_{t+h}^* \rangle$ that is aggregated as a new transition d_{t+h} in the dataset $\mathcal{D}^{i-1}$.

3. **Aggregation.** In order to collect training samples, LoOP exploits UCT_{LoOP} as an expert which – after each search – provides new transitions $\{d\}_{h=t}^{t+H}$ composing the $\mathcal{D}^i$ dataset. Such a dataset is aggregated into $\mathcal{D}^{0:i} = \mathcal{D}^i \cup \mathcal{D}^{0:i-1}$ as in [186, 185], and then, used to update the function approximator parameters $\boldsymbol{\theta}$.

4. **Estimate update.** The $\boldsymbol{\theta}$ parameters are used to update the Q_Φ value estimates. Then, the policy is generated in such a way that it maximizes $\pi_i(s) = \arg\max_a Q_\Phi(s, a)$.

Following, the next sections introduces two possible implementations of the Q-values function approximator Φ. However, it has to be highlighted, that the algorithm is designed to embrace any other implementation of Φ.

Function Approximator with Gaussian Mixture Model

To implement the function approximation Φ_{GMM}, we rely on previous literature [37] and we model Q-values using probability densities in the form of a mixture of G Gaussians [5]. GMMs provide the system with the flexibility, expressiveness and generality of non-parametric function approximators, while maintaining the benefit of estimating the uncertainty in the prediction. To update the Q-value estimates, we combine the approach in [5] with a data aggregation [186] procedure that exploits the dataset $\mathcal{D}^{0:i}$ collected at each iteration of LoOP in accordance with the following update rule

$$\begin{aligned} Q^i(s,a) = \; & Q^{i-1}(s,a) \\ & + \alpha(r + \gamma \max_{a'} Q^{i-1}(s', a') - Q^{i-1}(s,a)), \end{aligned} \tag{8.6}$$

Data: I the number of iterations; Δ initial state distribution; H UCT horizon; T policy execution timesteps, λ_0 initial max. Q threshold multiplier for admissible actions, $\epsilon_{\tilde{A}}$ probability for the UCT exploration term, α learning rate, γ discount factor.
Input: π_0 initial policy of the agent.
Output: π_I policy learned after I iterations.

begin
  for $i = 1$ **to** I **do**
    $s_0 \leftarrow$ random state from Δ.
    for $t = 1$ **to** T **do**
1) Get state s_t by executing $\pi_{i-1}(s_{t-1})$.
2) $\mathcal{D}^i \leftarrow \mathrm{UCT}_{\mathrm{LoOP}}(s_t, \lambda_0, \epsilon_{\tilde{A}})$.
3) $\mathcal{D}^{0:i} \leftarrow \mathcal{D}^i \cup \mathcal{D}^{0:i-1}$.
   $Q_\Phi \leftarrow \mathrm{UPDATE}(\mathcal{D}^{0:i}, \alpha, \gamma)$.
4) $\pi_i(s) \leftarrow \arg\max_a Q_\Phi(s, a)$.
    end
  end
  return π_I
end

Algorithm 2: LoOP

where s' and a' are respectively the state reached by applying a in s, and the action maximizing Q^{i-1} in s'. Q^{i-1} are the Q-values at previous iteration being $Q^0 = 0$. More in detail, the dataset is used to estimate a probability density function in the joint space of states and actions and Q-values.

The pdf depends on the set of parameters

$$\boldsymbol{\theta} = \{\pi_1, \mu_1, \Sigma_1, \ldots, \pi_G, \mu_G, \Sigma_G\}, \tag{8.7}$$

where π_i is the prior, μ_i the mean and Σ_i the covariance matrix of a Gaussians of dimensionality D in the canonical form

$$\mathcal{N}(\mathbf{x}, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)}. \tag{8.8}$$

The parameters $\boldsymbol{\theta}$ are estimated over the dataset $\mathcal{D}^{0:i}$, and they are obtained as the result of a standard Expectation-Maximization [48] procedure, initialized using the k-means algorithm [130] to avoid bad local optima. The number of components K of the GMMs is selected to minimize the Bayesian Information Criterion over testing portion of the dataset. Using $\boldsymbol{\theta}$, the pdf of a sample can be computed as

$$p(\mathbf{x}_i; \boldsymbol{\theta}) = \sum_{g=1}^{K} \pi_g \mathcal{N}(\mathbf{x}_i, \mu_g, \Sigma_g), \tag{8.9}$$

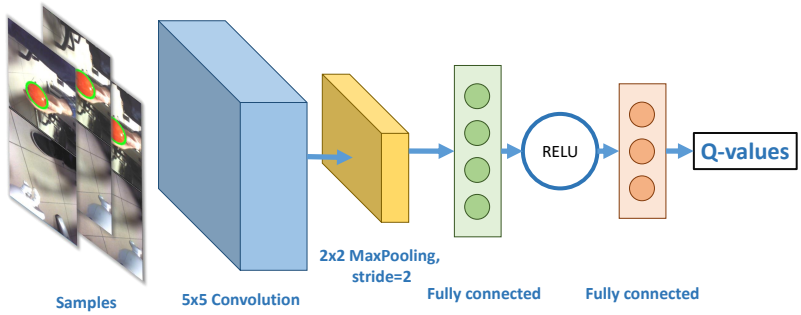


Figure 8.5. Deep convolutional neural network adopted in LoOP. The network is implemented within the MXNet environment.

while the value $\hat{Q}(s,a) \approx Q(s,a)$ is obtained as $\hat{Q}(s,a) = \mathbb{E}[Q|s,a] = \mu(Q|s,a)$. The approximated Q-value is the result of a Gaussian Mixture Regression

$$\hat{Q}(s,a) = \mu(Q|s,a) = \sum_{g=1}^{K} \beta(s,a)_g \mu_g(Q|s,a) \tag{8.10}$$

$$\sigma^2(Q|s,a) = \sum_{g=1}^{K} \beta(s,a)_g y_g(s,a) - \mu^2(Q|s,a), \tag{8.11}$$

that originates from the decomposition of each μ_g and Σ_g into

$$\mu_g = \begin{pmatrix} \mu_g^{(s,a)} \\ \mu_g^{Q} \end{pmatrix} \quad \Sigma_g = \begin{pmatrix} \Sigma_g^{(s,a)(s,a)} & \Sigma_g^{(s,a),Q} \\ \Sigma_g^{Q,(s,a)} & \Sigma_g^{Q,Q} \end{pmatrix}, \tag{8.12}$$

where $\mu_g(Q|s,a)$, $\sigma * 2_g(Q|s,a)$, $y_g(s,a)$ and $\beta_g(s,a)$ are respectively

$$\mu_g(Q|s,a) = \mu_g^Q + \Sigma_g^{Q,(s,a)}(\Sigma_g^{(s,a)})^{-1}((s,a) - \mu_g^{(s,a)}) \tag{8.13}$$

$$\sigma_g^2(Q|s,a) = \Sigma_g^{Q,Q} - \Sigma_g^{Q,(s,a)}(\Sigma_g^{(s,a)})^{-1}\Sigma_g^{(s,a),Q} \tag{8.14}$$

$$y_g(s,a) = (\sigma_g^2(Q|s,a) + \mu_g^2(Q|s,a)) \tag{8.15}$$

$$\beta_g(s,a) = \frac{\mathcal{N}(s,a;\mu_g^{(s,a)}, \Sigma_g^{(s,a),(s,a)})}{\sum_{i=1}^{K} \mathcal{N}(s,a;\mu_i^{(s,a)}, \Sigma_i^{(s,a),(s,a)})}. \tag{8.16}$$

Function Approximator with Deep Neural Networks

Deep learning shown a remarkable performance in learning effective policies in various Atari games [134, 135], where large state spaces and generalization made complex problems to be considered intractable. Motivated by recent advances achieved by DNNs, we choose to provide LoOP with the possibility to exploit different function approximators. This section presents the implementation of the Φ function as a deep neural networks.

In image classification [210] and RL applications [135], used DNNs feature a big number of layers, and thus, a tremendoues number of weights within. These models require a huge dataset of "good" training samples in order to converge

to acceptable solutions. However, in robotics, we do not dispose of such a huge dataset, and existing DNN models cannot apply. For this reason, as illustrated in Figure 8.5, we designed a minimal deep neural network (within the MXNet framework[1]), which has been implemented by stacking a convolutional layer and a max pooling layer with two fully connected layers, the former with ReLU activations while the latter with linears activations. The small and compact structure of the network adheres to our assumptions. In fact, it allows to efficiently compute reliable Q-value estimates even in robotic environment with few of training samples. In this configuration, $\boldsymbol{\theta}$ parameters of the function approximator are represented by the weights of the convolutional layers which are learned with the standard back-propagation algorithm [181, 75].

The Φ_{DNN} function assumes as input an image capturing the current state of the environment and generates a vector of Q-values, one for each action that the robot agent can perform. It is important to highlight that, we consider a sigle frame to contain all the information needed in a state s, and that we collect a dataset of transitions $\mathcal{D}^{0:i}$ where each element d is of the form $d = \langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$. Such a dataset is collected iteratively by aggregating new samples, using Eq. 8.2, during the robot operation [186]. Equivalently to replay buffer [134], iterative aggregation and random batch sampling, provide data decorrelation and facilitates learning and robustness. Operationally, batches are sampled until the dataset $\mathcal{D}^{0:i}$ is fully convered. Also in this case the dataset is used to minimize the loss-function (Eq. 8.3) and updates the Q-values estimates of the function approximator as in the GMM case

$$\begin{aligned} Q^i(s,a) = \; & Q^{i-1}(s,a) \\ & + \alpha(r + \gamma \max_{a'} Q^{i-1}(s',a') - Q^{i-1}(s,a)), \end{aligned} \tag{8.17}$$

8.2.2. Multi-agent planning

LoOP provides a flexible methodology that allows to learn action plans in complex environment such as in multi-agent settings. In this section, we show how LoOP naturally extends multi-agent scenarios, and in particular, we show its implementation in a set of fully collaborative games. To this end, we avoid to avoid to model joint actions [23, 40], thus, explosition of the state-space (and the UCT search complexity) by explicitly addressing problems where each agent has the same reward function, and that can achieve a common goal only by operating in cooperation. Our aim is to demonstrate that LoOP can learn competitive policies and constraint the search space both at the individual and collective level.

We extend the LoOP formulation under the stochastic game framework, as an extension of MDPs to multiple agent. In detail, it is represented as a tuple $(N, S, A_{1:N}, \mathcal{T}, R_{1:m})$, where N is the number of agents, $A_{1:N}$ is the cross product of the N agents' actions, and the other elements remain consistent to Eq. 8.1. Moreover, each agent j determines its actions in accordance with its policies π_j that defines its behaviors. Hence, each agent aims at improving its policy by maximizing the collective future reward. LoOP preserves the same procedural routine as introduced

[1] `https://mxnet.apache.org/`

Data: N number of agents; I the number of iterations; Δ initial state distribution; H UCT horizon; T policy execution timesteps, λ_0 initial max. Q threshold multiplier for admissible actions, $\epsilon_{\tilde{A}}$ probability for the UCT exploration term, α learning rate, γ discount factor.
Input: π_j^0 initial policy for each agent j
Output: π_j^I policy learned after I iterations

begin
  for $i = 1$ **to** I **do**
    $s_0 \leftarrow$ random state from Δ.
    for $t = 1$ **to** T **do**
1) Get state s_t by executing $\pi_j^{i-1}(s_{t-1})$ for each agent j.
      for $j = 1 \dots N$ **do**
2) $\mathcal{D}_j^i \leftarrow \mathrm{UCT}_{\mathrm{LoOP}}(s_t, \lambda_0, \epsilon_{\tilde{A}})$.
3) $\mathcal{D}_j^{0:i} \leftarrow \mathcal{D}_j^i \cup \mathcal{D}_j^{0:i-1}$.
        $Q_{j,\Phi} \leftarrow \mathrm{UPDATE}(\mathcal{D}_j^{0:i}, \alpha, \gamma)$.
4) $\pi_j^i(s) \leftarrow \arg\max_a Q_{j,\Phi}(s, a)$.
      end
    end
  end
  return π_j^I
end

Algorithm 3: LoOP - Cooperative

in Alg. 2. However, it extends to the multi-agent scenario by modifying two steps of the algorithm. It iteratively runs UCT for each agent, and it allows for simultaneous action execution whenever a policy roll-out is requested.

As detailed in Alg. 3 At each iteration i and timestep t, LoOP (1) runs – for each agent – an $\mathrm{UCT}_{\mathrm{LoOP}}$ search, by expanding actions according to their current Q-value estimates, and (2) collects new transitions d_j that are used to improve the Q-values of each agent. It stores agents sample separately (3) in order to allow for individual beahavior which is not effected by other experiences, and then, (4) maximizes Q-value estimates to generate an improved policy π_j^i.

It is worth highlighting that in the case $N = 1$, the algorithm presented in this section *simplyfies* to Alg. 2 as each of the LoOP features persist. Hence, in the rest of the chapter, we refer to as LoOP to the more general algorithm presented to address and represent multi-agent scenarios.

8.3. Optimistic Planning in Robotic Domains

The goal of this section is to validate our contribution over a set of different robotic applications. The experimental evaluation serves to highlight the main contributions5 of LoOP, namely *focused exploration* and *policy generalization*. The latter evaluated through different function approximators. Moreover, we demonstrate the robustness

of LoOP through the evaluation of its *meta-parameters*. The aim is to highlight the effectiveness of the algorithm, as well as its application to other robotic applications. Finally, we setup an experiment in a challenging scenario demonstrating that LoOP can be used to optimize local movements of complex behaviors. To this end, we task LoOP to learn the best position to assume for a defending robot in a multi-robot adversarial scenario. We evaluate the learned behaviors in RoboCup soccer games and report the performance improvement in terms of recovered balls.

All the experiments have been conducted using the V-REP simulator running on a single Intel Core i7-5700HQ core, with CPU@2.70GHz and 16GB of RAM. For all the scenarios, unless otherwise specified, the algorithm was configured with the same meta-parameters. The UCT horizon is set to $H = 4$ to trade-off between usability and performance of the search algorithm; the number of roll-outs is set to be $M = 3$, while admissible actions are evaluated with an initial $\lambda = 0.5$, and $\epsilon_{\tilde{A}} = 0.3$, guaranteeing good amounts of exploration. The C constant in Eq. 8.5 is set to 0.707. The state space, the set of actions and the reward functions are finally chosen and implemented depending on the robots and applications. In each of the proposed applications, stochastic actions are obtained by randomizing their outcomes with a 5% probability. We adopt a *shaped* reward function, thus providing a reward to the agent(s) at each of the visited states.

Throughout the experimental evaluation, we compare our approach against two baseline and two state-of-the-art algorithms. We compare LoOP with *vanilla-UCT* and *random-UCT* algorithms to show the improvements with respect to the policies generated by these standard routines. We refer to as *vanilla-UCT* as the standard UCT algorithm where each action is expanded at each execution of the MC search. Instead, we refer to as *random-UCT* as the UCT algorithm that selects an action at each time-step randomly. Then, we compare LoOP against TD-search [198] and DQN [134] – even though, a comparison with DQN can be done only in the case of Φ implemented as a DNNs. The experiments have been configured to highlight the performance improvement of LoOP, by comparing, for each of the evaluated algorithms, the cumulative average reward against the number of explored states.

8.3.1. Focused Exploration

The aim of this experiment is to show the benefits of a focused exploration, and thus, the effectiveness of LoOP in generating competitive policies already at first iterations. This is a key feature of our approach. It confirms that LoOP is practical, and can be exploited in robotics. To this end, a 7-DOF KUKA lightweight arm is tasked to learn to fetch an object (e.g. a glass, Figure 8.6) while avoiding an obstacle (e.g. a plant). In this scenario, the function approximator has been implemented as a DNN. Thus, the state is represented through an image collected by an overlooking camera, and the discretization of the state space is realized as described in Section 8.2.1. The robot can perform 10 actions: $A = \langle$`arm-up`, `arm-down`, `arm-forward`, `arm-backward`, `arm-right`, `arm-left`, `pitch-turn-left`, `pitch-turn-right`, `yaw-turn-left`, `yaw-turn-left`$\rangle$. Rotations on the roll angle have been removed as they do not influence the desired orientation of the fetched object. The reward function is in $[0, 1]$ and it is computed as a weighted sum of four components: the first is inversely proportional to the Euclidean distance of

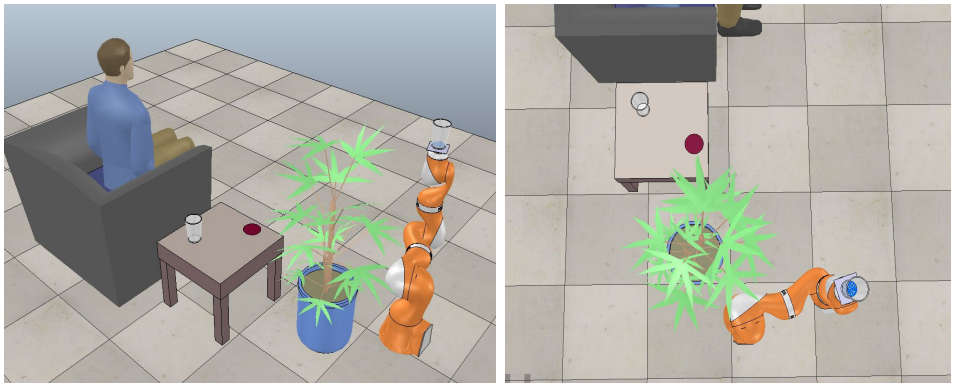
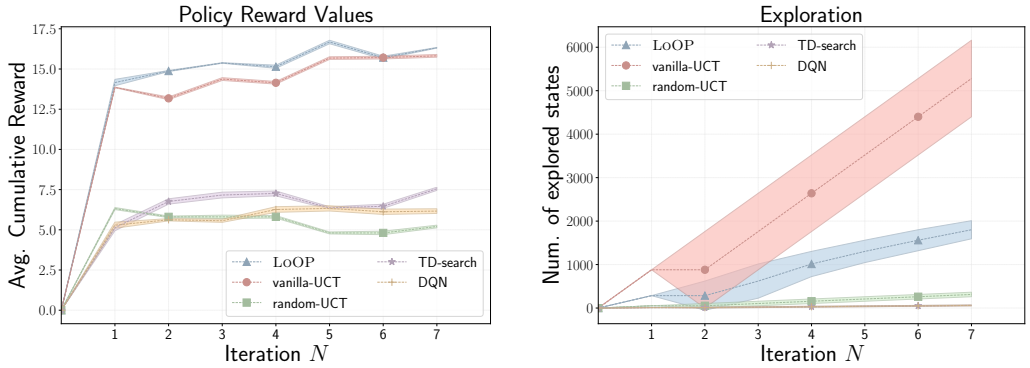


Figure 8.6. Bird-view (on the left) and top-view (on the right) of fetching task environment. The figure illustrates the position of the robot (7 DOF KUKA LBRiiwa arm), the obstacle (plant) and the target position of the object to fetch (red circle).



(a) Fetching task avg. cumulative reward

(b) Fetching task number of explored states

(c) Ending state after a roll-out of the LoOP policy at iteration 1

(d) Ending state after a roll-out of the DQN policy at iteration 1

Figure 8.7. Average cumulative reward and number of explored states obtained by LoOP, DQN, TD-search, *random-UCT* and *vanilla-UCT* in 7 iterations in the Kuka fetching scenario. For each of them, the reward is averaged over 10 runs. The function approximator has been implemented with DNN.

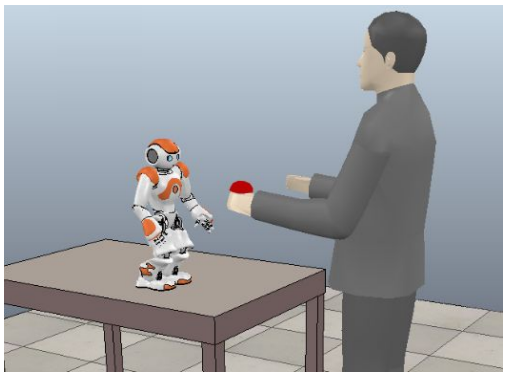
the end-effector to the target, the second it proportional to the distance to the virtual center of the obstacle, the third and the fourth are inversely proportional to the pitch and yaw angle respectively. In this way the reward function promotes states that are near the target, far from the obstacle, and with the end-effector oriented upwards. This is implemented to succeed in the fetching task of objects that are to be carried with a preferred orientation (e.g. a glass full of water). Finally, the function approximator is implemented (and learned) by using the deep neural network described in Section 4.

Figure 8.7a and Figure 8.7b show the performance of LoOP, DQN, TD-search, *random-UCT* and *vanilla-UCT*. For each of the iterations and for each algorithm, the continuous lines represent average cumulative rewards while the line width their standard deviation. In the explored state plots, the width of the lines highlights the amount of states expanded during i with respect to the total number of states explored until $i-1$. This scenario is key to our evaluation. In fact, since first iterations, and with a reduced set of training samples, LoOP is able to outperform other algorithms that need a huge dataset to learn competitive policies, e.g. DQN. Still, *vanilla-UCT* shows comparable rewards, but the number of explored states for this algorithm is ∼65% larger than LoOP. To provide a qualitative feedback on the benefits of the focused exploration, Figure 8.7c and Figure 8.7d show the *end states* reached by executing the policies learned by LoOP (Figure 8.7c) and DQN (Figure 8.7d) at the 1-*st* iteration of the algorithms. In particular, while the DQN policy initially explores portions of the state-space which do not lead to task completion, LoOP is able to "focus" the learning process since first iterations, and quickly achieve a good reward value. Such a result highlights that LoOP is more practical and thus a preferable solution in this setting.

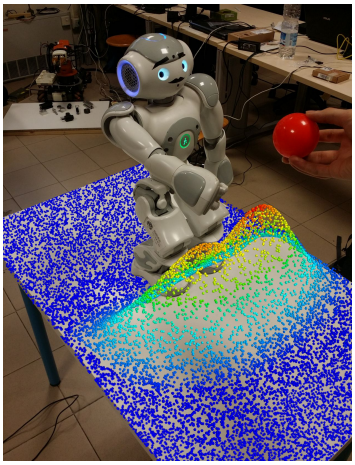
8.3.2. Policy Generalization

The second evaluation shows the performance of LoOP in learning an *handover* task with an hyper-redundant humanoid NAO robot. The aim is to show the ability of LoOP in generating competitive policy with a much complex platform, and its robustness in executing the learned policy on real platforms. We propose an evaluation of the reward signal by comparing (1) the two function approximators introduced in Section 8.2 – Φ_{DNN} and Φ_{GMM}, and (2) their performance on a real platform. The scenario is characterized by two agents performing a human-robot interaction. The robot has to complete an handover by receiving an object which is held by a human. As depicted in Figure 8.8a, we learn the action policy in the simulated environment, and then, we execute it on a real platform. Figure 8.8b qualitatively illustrates the *affordance semantic* distribution of the `right-arm-forward` action when following π_i in a state s_t, on the real platform.

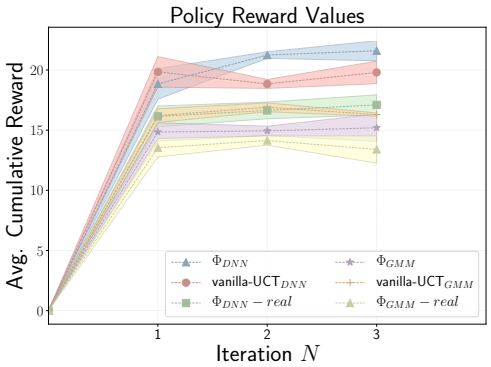
In this experiment, the agent can perform a set of 25 actions:
$A = \langle$ `body-noop`, `body-forward`, `body-backward`, `body-turn-left`, `body-turn-right`, `head-up`,`head-down`, `head-right`, `head-left`, `right-arm-up`, `right-arm-down`, `right-arm-left`, `right-arm-right`, `right-arm-forward`, `right-arm-backward`, `left-arm-up`, `left-arm-down`, `left-arm-left`, `left-arm-right`, `left-arm-forward`, `left-arm-backward`, `open-right-hand`, `open-left-hand`, `close-right-hand`, `close-left-hand` $\rangle$.



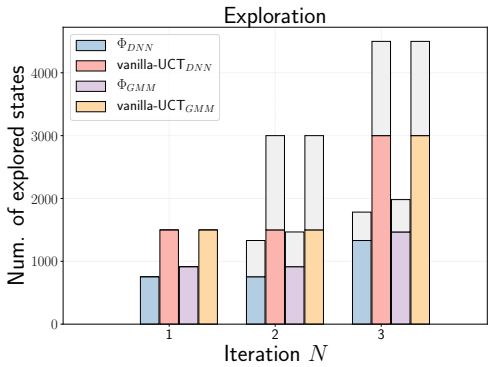
(a) Simulated environment for the Handover application



(b) Affordance semantics distribution of the `right-arm-forward` action in s_t



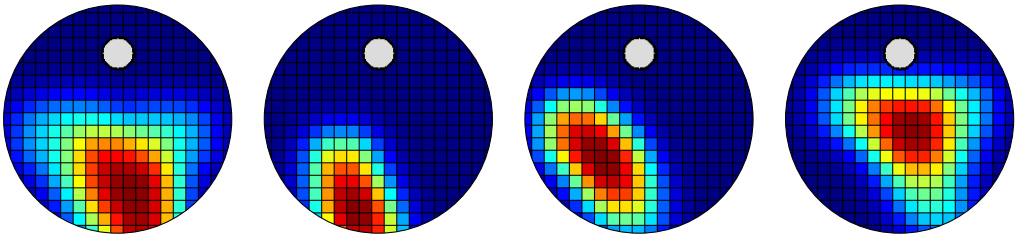
(c) Handover task avg. cumulative reward



(d) Handover task number of explored states

Figure 8.8. TEST DONE WITH DNNs.

The (shaped) reward function is in $[0, 1]$ and it is implemented as a weighted sum of 6 components: the first component is inversely proportional to the distance between the robot and the handed object, the second and third are inversely proportional to the distance from the object to the left and right hand respectively, the fourth component is inversely proportional to the distance between the ball and the center of the robot camera computed directly in the image frame, and the fifth and sixth components model the desired status of the hands, promoting states that with open hands near the handed object. Operationally, the state is represented differently depending on the implementation of the function approximator Φ. In the case of DNNs (Φ_{DNN}) the state is represented through the images collected by the cameras of the robot. Conversely, in the case of GMMs (Φ_{GMM}) the state is represented as a feature vector composed by 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). Moreover, the state includes the relative distance between the robot and object poses, the position of the target in the image frame of the cameras. Additional, a last feature element is included, the "attention bit", indicating if the human is looking towards the NAO. The "attention bit" is



(a) BODY-FORWARD **(b)** BODY-RIGHT **(c)** HEAD-RIGHT **(d)** HEAD-UP

Figure 8.9. Affordance semantics distribution over the state-space.

used to show-case the possibility to embed social rules into the robot behaviors.

Similarly to the previous experiments, Figure 8.8 shows the evaluation of the reward signal (Figure 8.8c) against the number of explored states (Figure 8.8d) for each function approximator. In the explored state plots, the gray top of each bar highlights the amount of states expanded during i with respect to the total number of states explored until $i-1$. As expected, LoOP preserves an improvement over the *vanilla-UCT* and, also in this case, our algorithm generates an effective policy with a remarkably reduced number of training examples. Differently, when comparing Φ_{DNN} against Φ_{GMM}, we notice a similar number of explored states. However, Φ_{DNN} obtains higher cumulative rewards. This shows the improved generalization capabilities of the DNN representation, that is able to significantly improve performance while preserving sample complexity of the original algorithm. Finally, Φ_{DNN}-real and Φ_{GMM}-real show the rewards obtained by transferring and rolling out the policy in real settings. In both cases, the reward values show that, even though the robot does not perform as well as in their simulated counterparts, it is still able to complete the task being robust to the noise of the real-world deployment. In fact, we consider the reduced cumulative reward values to be mostly due to noise in both the perception pipeline and motor encoders of the real NAO robot.

In addition to the learned policy, in the case of Φ_{GMM}, we qualitatively evaluate the obtained affordance semantics models. Figure 8.9 shows the heat-map – with a 5cm granularity – corresponding to the affordance semantics distribution of four actions: body forward (Figure 8.9a), body right (Figure 8.9b), head right (Figure 8.9c) and head up (Figure 8.9d). These have been generated from the model learned after 3 iterations of LoOP with a small initial distribution composed of states located at a distance between 45cm and 60cm in front of a the target object. From the figures it can be observed that the learned distribution accurately models the need of the robot to (1) go forward when it is far from the target and stop at a distance of 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 8.10 shows the Q-values 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 the chart, when the human is *not* paying attention (red bars), the highest

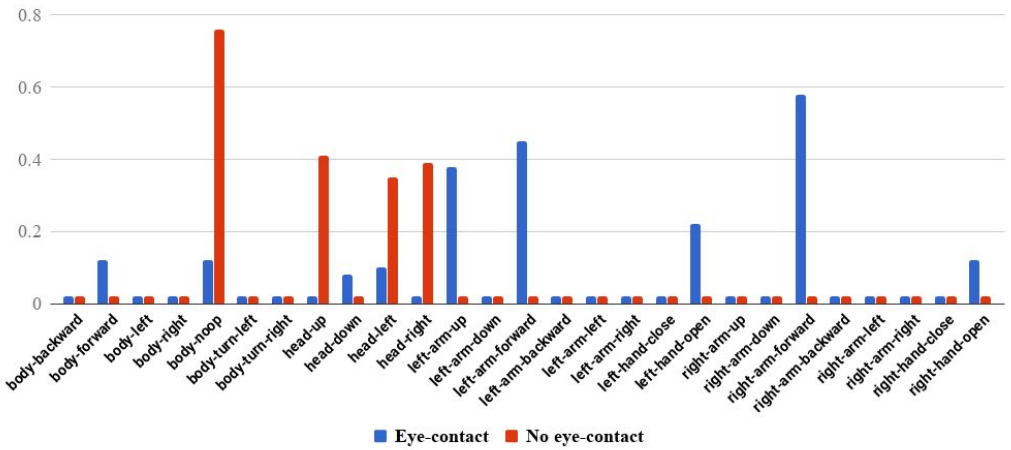
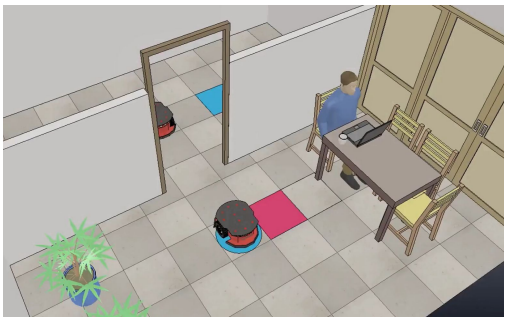
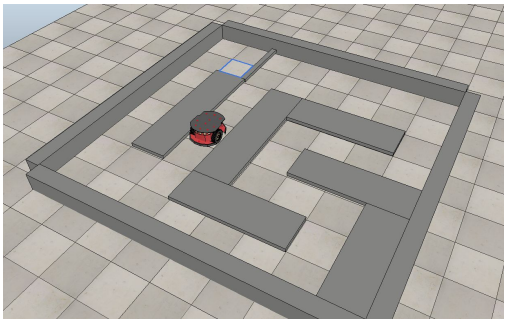


Figure 8.10. Q-value estimates for all the actions when the "eye contact" social rule is not respected and viceversa. The red bars corresponds to the Q-values estimates when the human does not pay attention to the robot, while the blue in the opposite case. Each action is labeled on the x-axis while on the y-axis its Q-value estimate.



(a) View of the door passing environment



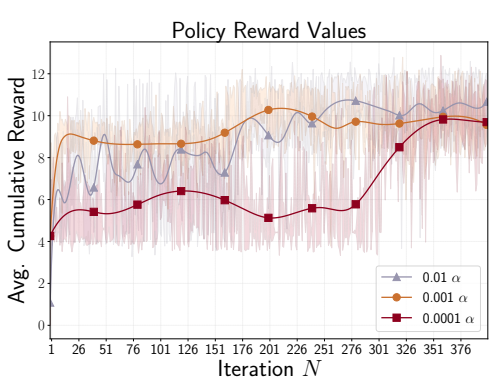
(b) View of the navigation environment

Figure 8.11. Environments used in the meta-parameters experimental evaluation for the two function approximators.

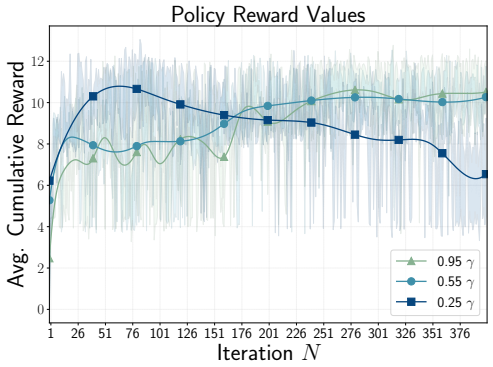
Q-value corresponds to the "null" action `body-noop`. Additionally, head movements are allowed to search for eye contact. Conversely, when the partner looks at the NAO (blue bars), the `right-arm-forward` action has the highest Q-value, and other arm-related actions are enabled. To embed this social behavior in the robot policy, we initialize the $\boldsymbol{\theta}_0$ 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.

8.3.3. Meta-parameters Evaluation

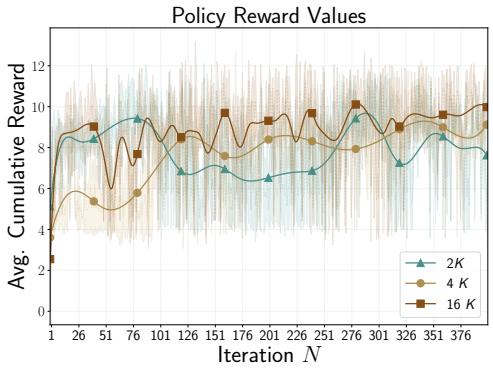
This set of experiments aims at validating the procedural routine of LoOP and its robustness. We propose a meta-parameters evaluation in two simulated robotic tasks implemented as grid worlds. The evaluation challenges LoOP in learning (1) a *door-passing* task – with a GMM-based function approximator – and (2) a



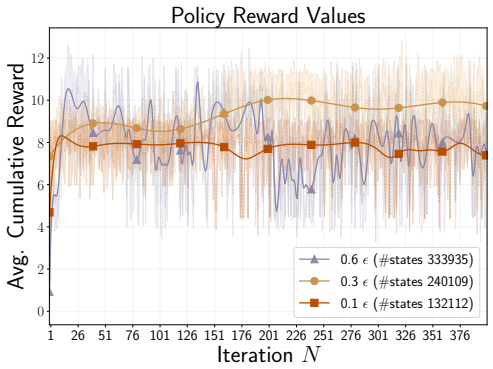
(a) Door passing learning rate α with GMMs



(b) Door passing discount factot γ with GMMs



(c) Door passing number of components GMMs K



(d) Door passing ϵ-probability in the UCT_{LoOP} expansion phase

Figure 8.12. LoOP performance in the door passing scenario.

path-finding task – with a DNN-based function approximator. Figure 8.11 illustrates the environments used to perform the meta-parameters evaluation, both in the case of GMMs (Figure 8.11a) and DNNs (Figure 8.11b). Equivalently, for the two scenarios, the robots can perform a discrete set of five actions $A = \langle$ `noop, up, down, right, left` $\rangle$ to move within the grid. The reward function is normalized between $[0, 1]$ and is shaped to be inversely proportional to the sum of the minimum path steps from the robot positions and targets. Differently, in the GMM case, the state is a feature vector composed by the current position of the robot and its assigned target. In the DNNs case, instead, the state is represented through an image collected from the top by an overlooking camera (i.e., all the robots are visible). To complete the evaluation, we vary the key meta-parameters of the algorithm such as the learning rate α; discount factor γ; ϵ-probability of the UCT_{LoOP} expansion phase; the number of GMMs components K; and number of the training runs of the DNN for each iteration tr. tr refers to the number of re-training refinements done at each iteration over the dataset D^i. We report the results of executing the learned policies on a Pioneer P3-DX robot.

Evaluation of the GMM Function Approximator

Specifically, in the case of the GMMs, the set of parameters is the learning rate α, discount factor γ, number of GMM components K and the ϵ-probability in the UCT_{LoOP} expansion phase. Throughout the experiments in Figure 8.12, the meta-parameters configuration is set to $\alpha = 0.001$, $\gamma = 0.55$, $K = 4$ and $\epsilon = 0.3$. Then, for each testing sessions, only one meta-parameter is changed at a time.

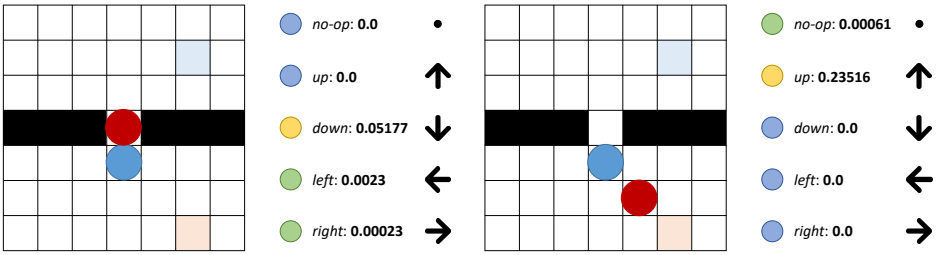
Figure 8.12a reports the learning curves by varying the learning rate α of Eq. 8.6. All the evaluated configuration are able to learn the task, however, with a different number of iterations. Noticeably, for the learning rate $\alpha = 0.01$ (purple), the reward signals significantly oscillates showing instability during training. Instead, $\alpha = 0.01$ (orange) presents the best trade-off between performances and stability of the learned values.

Similarly, Figure 8.12b reports the learning curves by varying the discount factor γ of Eq. 8.6. Both $\gamma = 0.55$ and $\gamma = 0.95$ are able to learn the task however, the latter configuration presents oscillations of the reward signals. For $\gamma = 0.25$ instead, the learner "forgets" the progress made at the beginning. This highlights the importance of the discount factor that weights expected future rewards. In fact, with small values of γ the robot is induced to show a more greedy exploration policy which favors local improvements rather than a competitive global solution for the task. Often, as reported here, this is the cause of a poor performance of the agent.

Figure 8.12c reports the learning curves by varying the number of GMM components. The figure shows that LoOP has the best performances with $K = 16$. Nevertheless, with a greater number of components the reward signal oscillates significantly, while with a lower values it poorly performs. This evaluation suggests the importance of choosing the K in accordance to the problem addressed. In fact, on the one hand, a bigger number of components can model complex non-linearities. But, it is also true that many components require a larger number of training samples to stabilize the reward signal. Moreover, the training time significantly increases with the number of components.

Finally, Figure 8.12d shows the learning curves by varying the ϵ-probability in the UCT_{LoOP} expansion phase (Eq. 8.4). It is worth highlighting that both with $\epsilon = 0.6$ and $\epsilon = 0.1$ the algorithm poorly performs, however, for different reasons. In fact, $\epsilon = 0.1$ significantly restricts the exploration of the algorithm and it is not able to visit valid portions of the state-space. $\epsilon = 0.6$, instead, allows for a more *in-breadth* exploration and requires the algorithm to visit a bigger number of times each state in order to generate satisfying reward values. Note that for $\epsilon = 1.0$ the algorithm shows the same exact routine of *vanilla-UCT*. For $\epsilon = 0.3$ the algorithm reports its best configuration and highlights the importance of a focused exploration also in this setting. In fact, in order to compare the configurations, the number of explored states is paired to the ϵ values in the figure. $\epsilon = 0.3$ shows the best trade-off between reward signals and number of explored states, and thus, computational time.

Figure 8.13 shows how the affordance semantics supports action planning of the blue robot in the critical state of the task. In the figures, the actions suggested by the policy are marked in yellow, while actions with small or zero Q-values are marked in green and blue respectively. Figure 8.13a in fact, shows that, when the robots



(a) Door passing state A **(b)** Door passing state B

Figure 8.13. Q-value estimate for different configurations of the state s_t for the blue robot. Actions, selected in accordance with the learned policy, are marked in yellow while actions with small or zero Q-values are marked in green and blue respectively.

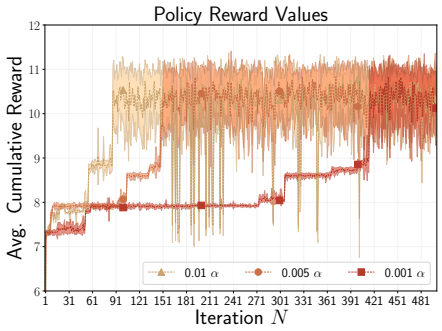
encounter an impasse, then the blue robot has converged to a less eager behavior and backtracks in order to let the red robot go first. Conversely, Figure 8.13b shows that, once the red robot has traversed the narrow passage, then the blue robot can proceed and complete the task. Both figures report the Q-values of each action in the two different configurations and show the correctness of the affordances semantics learned by LoOP.

Evaluation of the DNN Function Approximator

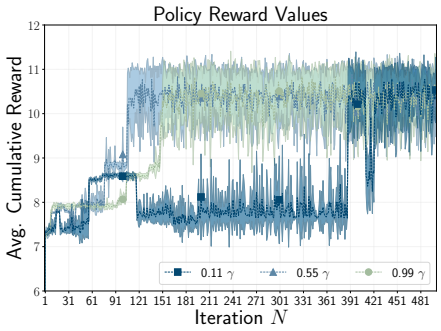
As in the case of Φ_{GMM}, the experiments in this section aim at evaluating the robustness of the proposed approach when varying the meta-parameters of the algorithm. In the case of the DNNs function approximator, the set of parameters is the learning rate α, discount factor γ, number of training runs for each iteration tr and the ϵ-probability in the UCT_{LoOP} expansion phase. In the experiments reported in Figure 8.14, the default meta-parameters configuration is set to $\alpha = 0.005$, $\gamma = 0.99$, $tr = 10$ and $\epsilon = 0.3$. For each testing session, then, only one meta-parameter is changed at a time.

Equivalently to the GMMs case, a similar analysis of the meta-parameters can be described. Figure 8.14a shows the learning curves by varying the learning rate in Eq. 8.17. The plot shows that the higher the learning rate the quicker the learner reaches competitive values, even though the reward signal of bigger α values reports larger oscillations. Figure 8.14b illustrates the evaluation of the discount factor (in Eq. 8.17) which, again, remarks the importance of the discount factor. In fact, also in this scenario, the learning curve of $\gamma = 0.11$ features drops in the task performance (e.g. around iteration $\sim$121 and $\sim$421).

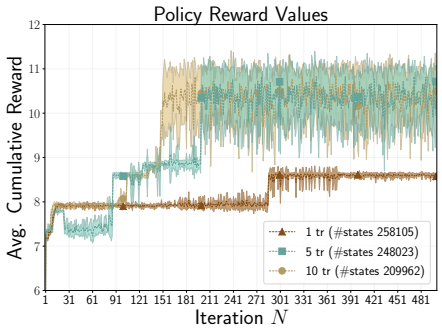
Figure 8.14c and Figure 8.14d report the learning curves by varying the number of training runs, and the ϵ-probability of Eq. 8.4. Additionally, the figures report the number of explored states paired with a specific value of the meta-parameters. It is worth noticing that, for lower values of the parameters the learner is not able to complete the task as it does not explores valid portions of the state-space. For higher values of the ϵ parameter, instead, the learner quickly reaches competitive values at the cost of training time – as shown by the number of explored states. Conversely, higher values of the tr parameters seems to support focused exploration and allows the agent to learn satisfying reward values before other the configurations.



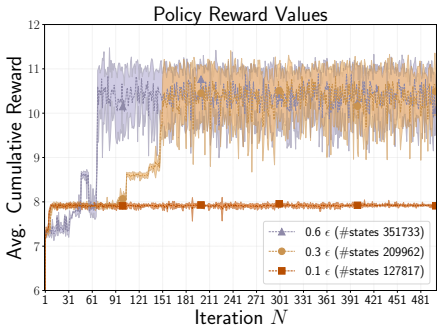
(a) Gridworld navigation learning rate α with DNNs



(b) Gridworld navigation discount factot γ with DNNs

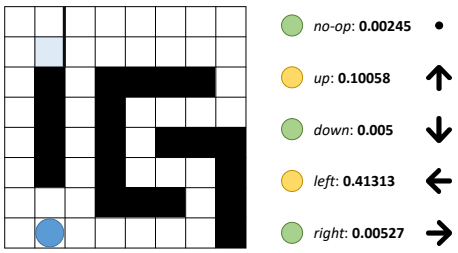


(c) Gridworld navigation number of train runs tr per iteration DNNs

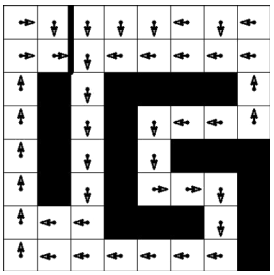


(d) Gridworld navigation ϵ-probability in the UCT_{LoOP} expansion phase

Figure 8.14. LoOP performance in the door passing scenario.



(a) Gridworld navigation state C



(b) Gridworld learned policy π_i

Figure 8.15. Q-value estimate for different configurations of the state s_t for the navigation task. Actions with a significant affordance semantics are marked in yellow, while actions with small$Q$-values are marked in green.

Finally, it is important to remark that DNNs are able to learn more stable policies and feature better generalization properties. In fact, their learning curves – despite the meta-parameters configuration – preserve a more stable and smooth profile, with smaller oscillations of the reward signal.

Figure 8.15 shows the correctness of the learned affordance semantics in the proposed task. Figure 8.15b illustrates the policy learned by the agent, while Figure 8.15a highlights a particular state of the task and the associated Q-values estimates. The Q-values show that, even if, the policy selects the `left` action to

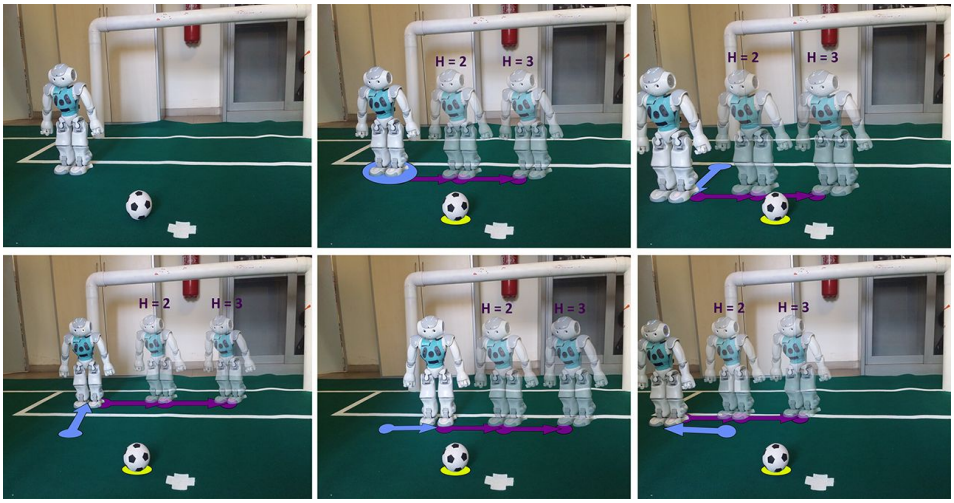


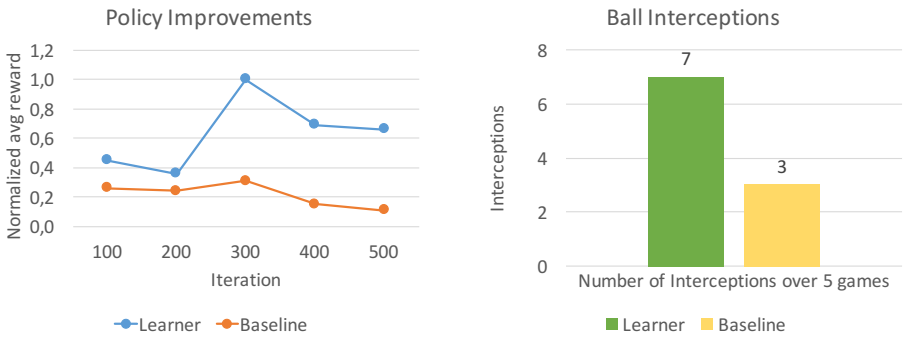
Figure 8.16. Example of a full iteration of the Monte Carlo roll-outs: the robot evaluates all its actions, and selects the best one to maximize $Q(s_t, a)$. In this example, the top-left figure shows the world state at a given time s_t, and the rollout policy commands the robot to execute the `move-left` action. Accordingly, the other sub-figures show the evolution of the system after each roll-out extending the current policy until the horizon $H = 3$. The robot evaluates all the 5 actions: `stand` (top-center), `move-up` (top-right); `move-down` (bottom-left); `move-left` (bottom-center); `move-right` (bottom-right). In these figures, the blue arrow represents the chosen action for the current roll-out, while the purple arrows represent the movements of the robot according to the current policy. The yellow circle represents the position of the ball.

move the agent, the other admissible action for LoOP is the `up` action. Such an action, in fact, equivalently supports task completion.

The meta-parameters evaluation conducted in this section highlights the robustness of LoOP with different algorithmic configurations and show-cases other two robotic applications where it applies. The stability of the reward signal varies upon parameters settings, however, LoOP can learn the a policy with a wide spectrum of such meta-parameters. This is an important result which enables the deployment of LoOP in different environments. For example, we can increase the learning rate α, and learn satisfying policies, when, in a particular scenario, the lack of training samples is a limiting condition.

8.3.4. Local Optimization of High-level Behaviors

LoOP can improve robot performance by optimizing local movements of of complex behaviors. In dynamic multi-agent adversarial scenarios, such as RoboCup competitions, robots have to feature the ability to perform long-term plans – to support team strategies – as well as short-term movements – to quickly react to environmental stimuli. In order to demonstrate the benefits of a local optimization of robot behaviors, we learn the positioning policy of a defending robot in soccer games. In this setting, we configure the state of the environment with the position of the robot and the position of the ball and its velocity vector. Then, the set of discrete robot actions is `stand` (the robot does not move), `move-up` (the robot moves



(a) Normalized avg. reward **(b)** Interceptions

Figure 8.17. On the right, the normalized average reward of the learner (*blue*) and baseline (*orange*) after different iterations. On the left, the sum of intercepted ball over five matches.

forwards), `move-down` (the robot moves backwards), `move-left`, `move-right`. The reward function is a weighted sum of two components. The former is the inverse of the Euclidean distance between the robot and the ball, while the latter is the inverse of the Euclidean distance between the robot and the line connecting the ball and the closest goal post. In this scenario, it is important to say that, LoOP learns Q-values without influencing the UCT expansion phase. To clarify the algorithmic procedure in this setting, Figure 8.16 shows the rollouts of the Monte-Carlo search for each actions. The top-left figure shows the world state at a given time s_t, and the rollout policy commands the robot to execute the `move-left` action. Accordingly, the other sub-figures show the evolution of the system after each rollout extending the current policy until the horizon $H = 3$. The robot evaluates all the 5 actions: `stand` (top-center), `move-up` (top-right); `move-down` (bottom-left); `move-left` (bottom-center); `move-right` (bottom-right).

The goal of our learner is to improve its performance while playing against opponent robots and to decrease the number of opponent scores while intercepting as many balls as possible. We can evaluate each action of our learner by considering the reward that the robot obtains during a match. Such a measurement expresses how good the learner is positioned within the field with respect to the ball. Therefore, we analyze the average reward of our agent as well as the number of ball interceptions and the final score of each match. Figure 8.17a reports the normalized average reward obtained by the learner during five regular games. On the y-axis is reported the obtained average reward.

Specifically, the learning defender features LoOP, while the non-learning defender has a fixed policy initialized at iteration zero. Such a baseline is a suitable comparison that allows us to quantify the improvements of our robot in terms of positioning with respect a baseline policy. The plot shows a constant improvement with respect to our baseline and over previous configuration of its trained policy. It is worth remarking that the drop in performance between game 3 and 4 can be due to different factors affecting the game, such as player penalization and ball positioning rules. However, such drop has a marginal impact with respect to the previous improvements, and

Table 8.1. The table reports the final scores of five matches after different MCSDA iterations.

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

the performance of consequent matches remains constant.

Additionally, thanks to the nature of our testing environment, we are able to report more direct evaluation indices for our approach. To this end, we report the number of intercepted balls and the number of opponent scores. In particular, Figure 8.17b shows the sum of intercepted balls of the two teams (learning and non-learning) on the same set of games as before, and Table 8.1 reports their final scores.

It is worth noticing that the number of intercepted balls of our agent (*green*) is more than twice the number of the opponent defender (*yellow*). Furthermore, the final results of the different matches promises an interesting profile: even though the learner does not win all of the matches, the number of opponent scores decreases as the learner refines its policy. Since LoOP is applied only on defense robots, we do not achieve any improvement on the number of goals of our team. However, as expected, by increasing the number of iterations of our algorithm, the number of goals of the opponent team decreases.

8.4. Concluding Remarks

In this chapter, we focus our efforts on an important component of the SK4R representation. We analyzed whether it is possible, for robots, to improve the representation of affordance semantics based on direct interaction with the environment. More precisely, we address the question:

4. *Given a representation of the state of the environment, can the robot improve its behavior over-time? In other words, is it possible to update (or learn) $\boldsymbol{\theta}$ parameters to improve the expressiveness of the AS representations?*

Our study confirms that it is possible to learn, and improve, the affordance semantics of actions through direct interactions with the environment – even in challenging robotic domains. In fact, our findings show that a functional representation of the environment can efficiently aid the robot in action execution. Hence, explicitly representing $\boldsymbol{\theta}$ parameters is beneficial, and it provides the robot with the opportunity to shape AS in accordance with the situation at hand.

To provide a efficient and practical example on the expressiveness of $\boldsymbol{\theta}$ parameters in representing ASs, we introduced LoOP, an iterative policy generation algorithm that uses action values to guide and reduce the exploration of the state space in different robotic applications. We have described its theoretical background and algorithmic procedure. We demonstrated its implementation through Gaussian Mixture Models and Deep Neural Networks, which exploit recent advances in deep

learning. We evaluated our approach over an extensive set of experiments which confirms the potential of LoOP in robotic applications.

Our key contribution is the combination of a Monte-Carlo tree search algorithm with Q-learning, which demonstrates a remarkable reduction of the computational load of the algorithm without loss in the performance. Such an improvement makes our approach more practical and suitable in difficult robotics applications, where the lack of training examples is often a limiting condition. We have shown the benefits of a focused exploration and of an effective policy generalization, which adds value to our solution. Additionally, we contribute to a modular algorithmic procedure that has been demonstrated to be flexible and practical in different scenarios.

However, LoOP presents several limitations that give the opportunity to further improve our solution. The major problem is the massive use of simulation calls, that make the algorithm less appealing. Hence, we are working on a online version by relying on online and offline knowledge as in [68]. Moreover, LoOP still needs a predefined simulation environment. This is not always available, and does not properly capture the dynamics of the world, making our algorithm less appealing in highly interactive scenarios. To address this issue, we aim at learning the dynamics of the world at robot operation time, and simultaneously improving its policy based on the learned model [228]. Finally, our solutions exploits discrete set of atomic actions. Hence, we want to extend our formulation to learn policies in the continuous action space [135] in order to address a broader set of robotic tasks.

Chapter 9

SK4R for Hierarchical Optimistic Planning

For a cognitive robot, task decomposition is key to guarantee effectiveness in various applications where the global complexity of a problem makes planning and decision-making too demanding. This is true, for example, in different domains, where unpredictabilities and modeling limitations typically prevent (1) the design of robust behaviors, and (2) learning effective action policies. To tackle this problem, we enhance the formulation of LoOP, of the previous chapter, to hierarchical optimistic planning. We borrow the concept of Hierarchical Task Networks (HTNs) to decompose the learning procedure, and we exploit Upper Confidence Tree (UCT) search to introduce h-LoOP, an iterative algorithm for hierarchical optimistic planning with learned value functions. h-LoOP relies upon the same working principles of LoOP but, instead of focusing on the κ-function parameters module (SK4R$_P$), it is also used to learn the hierarchical structure of the κ-function within the SK4R representation. In this chapter, we describe the h-LoOP algorithm by highlighting the key algorithmic differences with respect to LoOP (Section 9.1). Then, Section 9.2 describes two robotic applications that we use to setup the experimental evaluation of a fetching task using a simulated 7 DOF KUKA arm and, on a pick and delivery task with a Pioneer P3-DX robot. Finally, Section 9.3 concludes the chapter by discussing its key features.

9.1. Hierarchical Optimistic Planning

As in Chapter 8, here, we focus on the problem of robot planning by combining UCT Monte-Carlo search with Q-learning. However, we enhance LoOP by formalizing a hierarchical algorithm that decomposes complex tasks and further improves exploration at learning time.

In literature, multiple authors exploit the notions of skills and semi Markov Decision Processes (semi-MDPs), and define hierarchical representations such as *options* [208] and MAX-Q decompositions [54]. Unfortunately, applications of these methods in complex domains like robotics are limited, and prior knowledge has to be enforced in the learning process by means of expert demonstrations. In fact, although hierarchical learning and value function approximations techniques

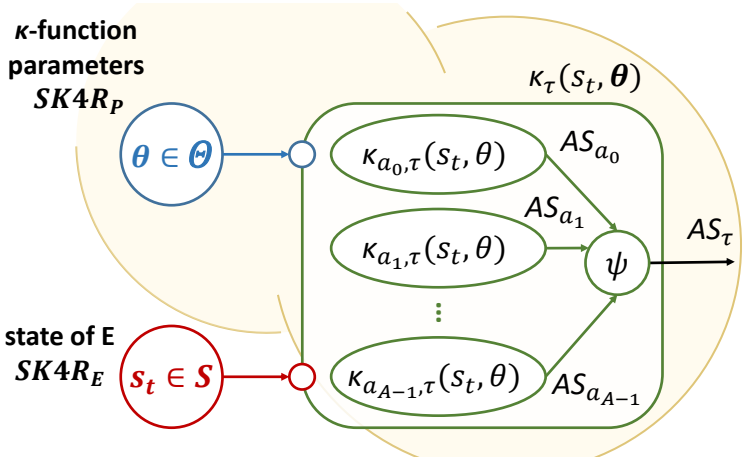


Figure 9.1. The κ-function parameters module SK4R$_P$ and the κ-functions.

have been adopted in several applications, state-of-the-art approaches still show considerable margin of improvement. For example, [192] provide a better policy generalization by exploiting the concept of Generalized Value Functions, to improve value function approximation.In a different settings, [39] use expert demonstrations to learn high-level tasks as a combination of action-primitives. Unfortunately, these approaches only learn specific hierarchical structures, that poorly generalize and cannot profit from the expressiveness of value functions. Similarly, [206] apply hierarchical learning to sequences of motion primitives on a pick-and-place task with a hyper-redundant robotic arm. [109] initialize skill trees from human demonstrations, improving them over time. However, their representations use expert demonstrations and do not represent action on higher levels of abstractions. Conversely, [94] apply hierarchical policy learning to solve a 2-DOF stand-up task for a robotic arm. They exploit Q-learning and actor-critic methods to learn both task decompositions and local trajectories that solve specific sub-goals. Alternatively, [89] and [6] formalize action hierarchies to represent actions at different levels of abstraction. However, these procedures are not easily scalable to higher dimensionality problems.

In contrast in this chapter, we adopt a hierarchy of actions to make state-of-the-art planning algorithms practical in robotic settings. To this end, we introduce h-LoOP [35], an iterative algorithm for learning hierarchical value functions, that are used to (1) capture multi-layered affordance semantics, (2) generate policies by scaffolding the acquired knowledge, and (3) guide the exploration of the state space. As in the case of LoOP, the goal of h-LoOP is to learn κ-parameters (i.e. $\boldsymbol{\theta}$) to shape affordance semantics of robot actions, however, it also learns hierarchical relations among κ-functions. It is worth remarking that, in previous chapters, the design of κ-function relations and interactions is decided before robots deployment. Accordingly, Figure 9.1 recalls the overall SK4R representation and highlights (in yellow) the subject of study of this chapter: (1) how to learn $\boldsymbol{\theta}$ parameters for robot planning and (2) how to learn a hierarchical representation of robot actions to enable task decompositions.

More in detail, h-LoOP improves LoOP by building upon concepts from previous literature, such as Hierarchical Task Networks (HTNs) [60], semi-MDPs [208] and

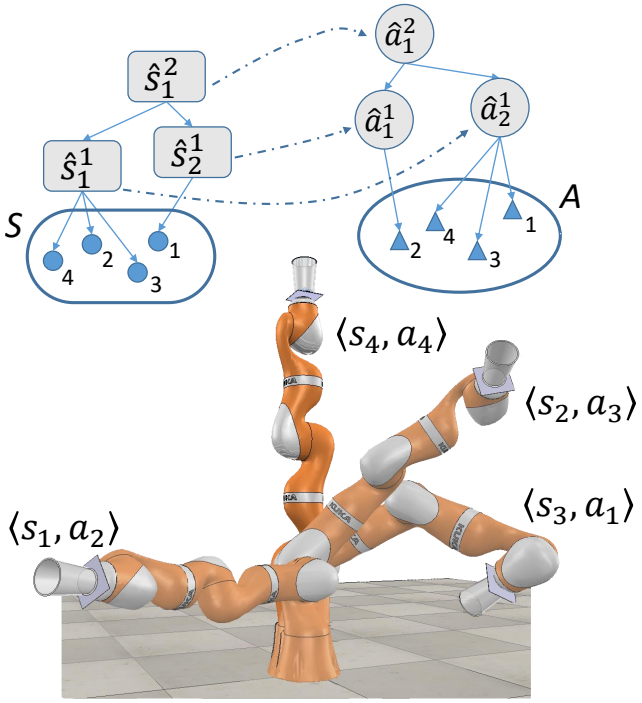


Figure 9.2. *h*-LoOP generates high-level representations of actions, that are used to improve the exploration of the search space. In this figure, we show the action hierarchy generated for a fetching task using a redundant KUKA light weight arm.

MAX-Q decompositions [54], to decompose the learning procedure and to generate both action abstractions and search space constraints. The action hierarchy formalized by *h*-LoOP is learned iteratively by evaluating state-actions pairs generated by UCT after each episode. Figure 9.2 shows an example of such a hierarchy where states and actions are associated at different layers of abstraction. *h*-LoOP assigns states and actions to different clusters s_l^c and a_l^c by evaluating the similarity of successor states that the agent can reach, by applying the actions in a_l^c in the states contained in s_l^c. Intuitively, similar successor states have similar reward values and can be evaluated altogether when exploring the search space. Different layers provide different granularity of affordance semantics (the higher the more coarse) and help the learning process to evaluate states hierarchically. *h*-LoOP runs UCT to explore the environment by sampling the joint distribution of rewards and state-action pairs. Each sample is continuously aggregated into a dataset, that is used to estimate – by means of Q-learning – the value function Q^λ of each layer λ in the hierarchy. Specifically, at each layer, Monte Carlo search is ran for a subset of actions that are evaluated according to their Q-value, thus driving the node-expansion phase during episode simulation.

In this chapter, our aim is to demonstrate that Q-values can be learned hierarchically to influence exploration, and to represent affordance semantics at different levels of abstraction, thus linking learning techniques to low-level agent controls. The main contributions of *h*-LoOP consists in: (1) a novel integration of Monte-Carlo tree search, hierarchical decision-theoretic planning and Q-learning, that enables

good performance with selective state exploration and improved generalization capabilities; in (2) a two-sided extension of TD-search, that not only executes on multiple hierarchy layers, but also constructs upper confidence bounds on the value functions – and selects actions optimistically with respect to those; and in (3) a reduction of the curse-of-dimensionality that is obtained by means of focused exploration. We evaluated the h-LoOP performance in two different scenarios, an *fetching* task with a 7-DOF KUKA light weight arm (as in Chapter 8) and, a *pick and delivery* task with a Pioneer P3-DX robot, where the agent has to collect an item and delivering it. The results show a reduction in the number of states explored – which makes the method more practical in robotics – and, the ability of h-LoOP to learn affordance semantics of robot actions and their hierarchical relations.

9.1.1. h-LoOP

In respect to the LoOP formulation, we describe h-LoOP by adopting the Markov Decision Process (MDP) notation, in which the decision-making problem is represented as a tuple $MDP = (S, A, \mathcal{T}, R, \gamma)$, where S is the set of discrete states of the environment, A represents the set of discrete actions, $\mathcal{T} : S \times A \times S \rightarrow [0,\ 1]$ is a stochastic transition function that models the probabilities of transitioning from state $s \in S$ to $s' \in S$ when taking action $a \in A$, $R : S \times A \rightarrow \mathbb{R}$ is the reward function, and γ is a discount factor in $[0, 1)$. 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.

Given an MDP, the goal consists in finding a policy $\pi(s)$ that maximizes its expected cumulative reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$ over a finite or infinite time horizon T. This can be obtained by using the notion of action-value function $Q^\pi(s, a)$ and by solving its corresponding Bellman optimality equations [208]

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} \mathcal{T}(s, a, s') \max_{a'} Q^*(s', a').$$

In this chapter, we address the generalization problem with Φ_{GMM} by approximating the Q function using probability densities in the form of a mixture of K Gaussians (i.e., Gaussian Mixture Models – GMMs), with K determined in an adaptive manner [5]. However, nothing prevents the implementation of h-LoOP with DNN, or an equivalent function approximator. Accordingly, $\boldsymbol{\theta}$ parameters are learned and used as described in Section 4 in Chapter 8 (Page 118).

h-LoOP is an iterative algorithm that, at each iteration i, generates a new policy π_i which improves π_{i-1}. To obtain an improved π_i, our algorithm leverages (1) data aggregation [186], and (2) UCT [26] for balancing between exploration and exploitation on the tree.

Exploration and Sample Collection

At every step h, for $h = 1 \dots H$, UCT simulates the execution of all the actions $\tilde{\mathcal{A}}_s \subseteq \mathcal{A}$ that are "admissible" in s_h, as detailed in next section. Each simulation executes an action $a \in \tilde{\mathcal{A}}_s$, followed by K roll-outs, that run an ϵ-greedy policy based

on π_{i-1} until a terminal state is reached. The best action a_h is selected according to

$$\begin{aligned} e &= C \cdot \sqrt{\frac{\log(\sum_a \eta(s_h, a))}{\eta(s_h, a)}} \\ a_h^* &= \arg\max_a \hat{Q}(s_h, a) + e, \end{aligned} \tag{9.1}$$

where C is a constant that multiplies and controls the exploration term e, and $\eta(s_h, a)$ is the number of occurrences of a in s_h. Since we assume a discrete state space $\mathcal{S}$, for continuous problems we define a similarity operator that informs the algorithm whether the difference of two states is smaller than a given threshold ξ – thus discretizing the space.
During each roll-out:

- a dataset $\mathcal{D}^i$ of samples $x = (s, a, Q^i(s, a))$ is collected to improve our estimate $\hat{Q}$, as detailed in Section 4 in Chapter 8;
- h-LoOP uses UCT as an expert and collects a dataset $\mathcal{D}^i$ of H samples $x = (s_h, a_h^*, s_h')$ that are selected by the tree search
- similarly to DAgger [186], $\mathcal{D}^i$ is aggregated into a dataset $\mathcal{D}^{0:i} = \mathcal{D}^{0:i-1} \cup \mathcal{D}^i$.

When H UCT steps are run, the complete dataset $\mathcal{D}^{0:i}$ is used to generate hierarchy clusters – as detailed in the next section – and learn a new policy π_i.

9.1.2. Hierarchical Action Selection

The hierarchical model adopted in h-LoOP builds upon the concepts of *High-Level Actions (HLAs)* and *Reachable Sets (RSs)* in HTNs [60].

- HLAs are defined recursively as a sequence of action primitives and/or other HLAs. When a HLA is composed by only primitives, such sequence is called "implementation".
- RSs model preconditions and effects of HLAs. They are defined as the union of possible states reachable by the the different implementations of HLAs. A RS is bounded by a pessimistic RS^- and optimistic RS^+ set. RS^- represents the set of states that are reached independently from the chosen implementation, while RS^+ represents the set of states reached by all the possible implementations. Reachable sets describe interesting properties: (1) if RS^- intersects the set of goal states, then a sequence of admissible actions leading to the goal is found; (2) if, instead, RS^+ intersects the set of goal states, a plan exists but its implementations do not reach the goal yet.

Using similar concepts, we allow h-LoOP to evaluate actions at multiple levels of abstraction. In order to better describe the action hierarchy generated by h-LoOP, Figure 9.3 illustrates a simplistic example. Specifically, a hierarchy of actions is obtained using an *agglomerative clustering* algorithm, which is ran on the set of next states $\{s'\}$ present in the $\mathcal{D}^{0:i}$. Our key assumption is that similar s' encode information about actions with similar effects and thus can be clusterized altogether.

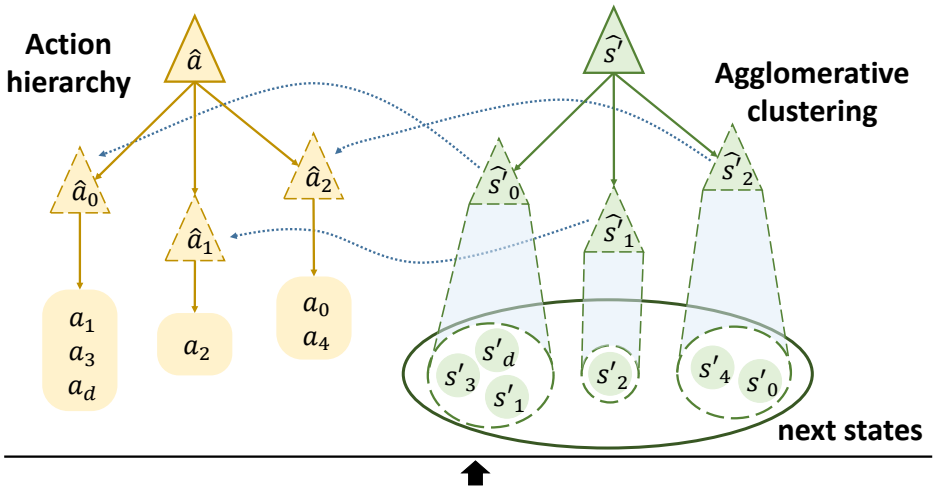


Figure 9.3. Simplistic example of action clusters generated by h-LoOP. The next states s' existing in the complete dataset $D^{0:i}$ (at iteration i) are agglomerated in a predefined number of clusters. The structure of the generated hierarchy of states is preserved to define a second hierarchy $\mathcal{H}$ dedicated to actions. In fact, actions a, associated to s' in $D^{0:i}$, are arranged along $\mathcal{H}$ by retracing actions of $D^{0:i}$, whose next states s' have been clustered together.

Particularly, we refer to $\mathcal{H}$ as the set of layers in the action hierarchy generated by the clustering algorithm. The clustering routine organizes $\{s'\}$ in clusters $\hat{s}'$ over a predefined number of layers, which are then tranferred into the action space, generating a set of action clusters $\hat{a}$ organized along the same structure. Such a mapping is realized by evaluating each element contained in the $\hat{s}'$ clusters and backpropagated to the original dataset $\mathcal{D}^{0:i}$ in order to retrieve the set of actions generating the transitions to the $\{s'\}$ states. In fact, dataset elements are a tuple of three components encoding a transition from the current state s to the next state s' by means of the action a.

It is important to highlight that, in this setting, it is no possible to clusterize actions directly, since actions are referred to at the symbolic level (e.g. `move-arm`, `turn-body`). Conversely, states are represented as feature vectors and it is possible to define a distance operator to run the clustering algorithm. Moreover, the learner cannot observe similarity among actions by means of their symbolic label. It is intuitive, instead, to compare similarities of the effects of robot actions by observing the next states within the dataset $\mathcal{D}^{0:i}$.

In our formulation, each action cluster $\hat{a}$ corresponds to a HLA in the layer $\lambda \in \mathcal{H}$, and each layer has an associated Q^λ function, approximated as $\hat{Q}^\lambda$. The result of choosing an action $\hat{a}$ consists of selecting a cluster of lower level actions with a similar expectation to reach a desired cluster $\hat{s}'$. Noticeably, such a model intrinsically connects to the concept of reachable sets. Clusters $\hat{s}'$ are in fact an approximation of optimistic sets RS^+, and they evaluate actions $\hat{a}$ that lead to more rewarding states altogether.

h-LoOP uses each Q^λ in s to select the set $\tilde{\mathcal{A}}_s$ of admissible actions for UCT.

Intuitively, a primitive action a is admissible in s if, for each layer λ, a belongs to the cluster $\hat{a}^\lambda$ selected according to Q^λ. More formally:

$$\tilde{\mathcal{A}}_s = \bigcup_{a \in \mathcal{A}} \{a \mid \forall \lambda \in \mathcal{H} : a \in \hat{a}^\lambda\} \tag{9.2}$$

$$\hat{a}^\lambda = \arg\max_{\hat{a}} Q^\lambda(s, \hat{a}) + \delta^\lambda_{s,\hat{a}} \tag{9.3}$$

$$\delta^\lambda_{s,\hat{a}} \sim \mathcal{N}(0, \sigma^2(Q^\lambda \mid \ s, \hat{a})),$$

where σ^2 is the standard deviation of the regression approximation [5].

Through $\delta^\lambda_{s,\hat{a}}$, the prediction error for each action abstraction is captured, leading to a more directed exploration of the action hierarchy. Action primitives are finally chosen and executed by UCT according to the lowest-level $\hat{Q}$, as detailed in previous section. To obtain a less biased exploration and avoid value function over-fitting, "inadmissible" actions are anyway expanded and selected by UCT with a 30% probability.

9.1.3. *h*-LoOP Algorithm

The goal of h-LoOP consists of iteratively updating each layer's value function approximation $\hat{Q}^\lambda$, to generate a policy π_i that maximizes the expected reward of the agent. The underlying insight of h-LoOP is that, while exploring the search space, collected state-action pairs are used at each iteration i to (1) update the approximated Q functions for refining the policy π_{i-1} into a policy π_i, and (2) use Q-values to influence UCT exploration in accordance with Eq. 9.2. The complete h-LoOP algorithm – described in Alg. 4 – adopts a procedure similar to LoOP, however it enhances it as follows:

1. **Roll-in.** The agent follows the previous policy π_{i-1} and generates a set of s_t states for T timesteps.

2. **UCT search.** For each of the generated states s_t, h-LoOP runs an UCT search with horizon H. At each step h, UCT simulates the execution of every admissible action in the set $\tilde{\mathcal{A}}_s$, computed according to Eq. 9.2. For each action $a \in \tilde{\mathcal{A}}_s$, a simulation consists of the execution of a, followed by K ϵ-greedy roll-outs based on π_{i-1}, which are used to estimate the Q-values of each visited state. Finally, for each step, the best action a^*_h is (1) chosen according to Eq. 9.1 and (2) aggregated into a dataset D^i together with s_h and s_{h+1}. It is worth remarking that a vanilla implementation of the UCT search evaluates all possible actions and explores a significant amount of states to generate an effective policy. Our approach, instead, leverages the hierarchical structure of $\mathcal{H}$ to generate a restricted subset of admissible actions, with high estimated Q-value. This efficiently reduces the exploration phase by guiding the algorithm to discard actions that are not expected to improve π_{i-1}.

3. **Hierarchical data aggregation and $\hat{Q}$ update.** After UCT, new data $D^i = \{(s_{t+h}, a^*_{t+h}, s'_{t+h+1}) \mid h = 1 \dots H\}$ is available to be aggregated into a larger dataset $\mathcal{D}_{\pi,i}$. This dataset is used to generate clusters $\hat{s}$, $\hat{a}$, and $\hat{s'}$ in two steps: first, the sets of states $\{s\}$ and $\{s'\}$ in $D_{\pi,i}$ are separately clustered

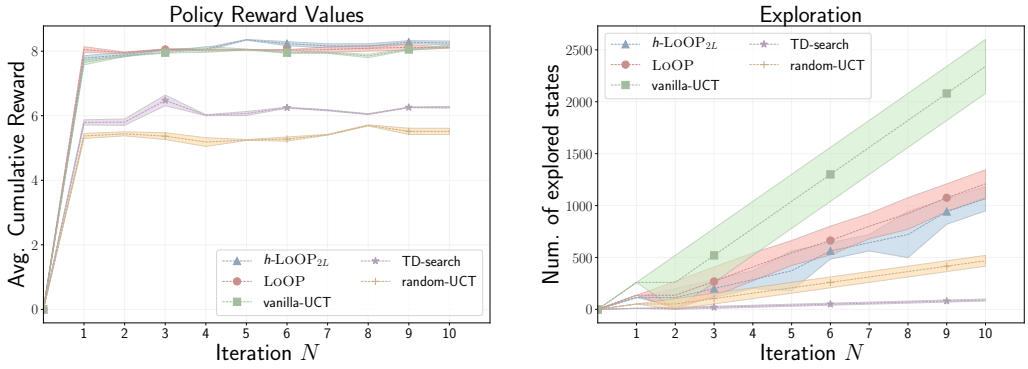
Input: $\mathcal{D}^0$ dataset of random state action pairs $\{(s, a, s')\}$.
Output: π_N policy learned after N iterations of the algorithm.
Data: A set of primitive actions, N number of iterations of the algorithm, Δ initial state distribution, H UCT horizon, K ϵ-greedy roll-outs, T policy execution timesteps, $\mathcal{H}$ set of layers.

begin
  Initialize $\hat{Q}^0$ to predict 0.
  Train classifier π_0 on $\mathcal{D}^0$.
  for $i = 1$ **to** N **do**
    $s_0 \leftarrow$ random state from Δ
    for $t = 1$ **to** T **do**
      Get state s_t by executing $\pi_{i-1}(s_{t-1})$.
      $\mathcal{D}^i \leftarrow \text{UCT}(\mathcal{H}, s_t)$
      $\mathcal{D}^{0:i} \leftarrow \mathcal{D}^{0:i-1} \cup \mathcal{D}^i$
      $\hat{s}, \hat{s}' \leftarrow$agglomerativeClustering($\mathcal{D}^i$)
      $\hat{a} \leftarrow$ mapping($\hat{s'}$)
      foreach $\lambda \in \mathcal{H}$ **do**
        // update estimated Q-values
        $\bar{R}(\hat{s}, \hat{a}) \leftarrow \frac{1}{|(\hat{s},\hat{a})|} \sum_{s \in \hat{s}, a^* \in \hat{a}} R(s, a^*)$
        $\mathcal{D}^{\lambda,i} \leftarrow$ getSamples($\mathcal{D}^i$, $\bar{R}(\hat{s}, \hat{a})$)
        $\mathcal{D}^{\lambda,0:i} \leftarrow \mathcal{D}^{\lambda,0:i-1} \cup \mathcal{D}^{\lambda,i}$
        Train $\hat{Q}^{\lambda,i}(\hat{s}, \hat{a})$ on $\mathcal{D}^{\lambda,0:i}$
      end
    end
    Train classifier π_i on $\mathcal{D}^i$
  end
  return π_N
end

Algorithm 4: h-LoOP

within λ layers; then, the hierarchy of next state clusters $\hat{s'}$ is transferred into the action space to generate the action clusters $\hat{a}$, each of them corresponding to high-level actions. In order to correctly update the $\hat{Q}^\lambda$ estimation for every $\hat{a}$, samples of the form $\mathbf{x}^\lambda = (\hat{s}, \hat{a}, Q^\lambda(s, \hat{a}))$ are generated for each layer λ, with $Q^\lambda(s, \hat{a})$ determined through Φ_{GMM}. Such samples are then (1) aggregated into a dataset $\mathcal{D}^{\lambda,0:i}$ and (2) used to improve the estimate $\hat{Q}^\lambda$, as described in previous sections. Specifically, $\hat{Q}^\lambda$ is updated for each $(\hat{s}, \hat{a})$ containing the state-action pairs $(s_{t+h}, a^*_{t+h}) \in \mathcal{D}_{uct}$, by using the averaged reward $R(s_{t+h}, a^*_{t+h})$ of the corresponding state-action pairs in the clusters $(\hat{s}, \hat{a})$.

4. **Training.** Once data aggregation has been performed, a new policy π_i is trained from the dataset $\mathcal{D}^i$.



(a) avg. cumulative reward **(b)** number of explored states

Figure 9.4. Average cumulative reward and number of explored states obtained by *h*-LoOP$_{2L}$, LoOP, TD-search, *random-UCT* and *vanilla-UCT* in 10 iterations in the Kuka fetching scenario. For each of them, the reward is averaged over 10 runs. The function approximator has been implemented with GMMs.

9.2. *h*-LoOP Experimental Evaluation

We evaluate our approach in learning an effective policy for executing a *fetching* task, and a *pick and delivery* task in a simple environment [54], with a reduced number of state-action pairs. We compare our results with a random-UCT and vanilla-UCT implementations, the TD-search [198] algorithm and different configurations of the *h*-LoOP action hierarchy. We will refer to as LoOP as a basic implementation of the action hierarchy composed solely by a single layer of the primitive actions. Then, the random-UCT algorithm selects a random action at each h step of the Monte Carlo search, while in the vanilla-UCT the algorithm expands all the robot actions at each step. In all algorithms, we implement a shaped reward function that computes reward values at each visited state. We deploy these algorithms within a simulated environment on a 7 DOF KUKA arm and, on a Pioneer robot.

Experiments have been conducted within the V-REP simulation environment, using a single Intel i7-5700HQ core, with CPU@2.70GHz and 16GB of RAM. For both the scenarios, the UCT search has been configured as follows: (1) search horizon $H = 4$; (2) exploration constant $C = 0.707$; (3) $K = 3$ roll-outs. The number of components of the GMMs is evaluated according to the BIC criterion, which has been tested using up to 6 Gaussians. Q-values are updated with a learning rate $\alpha = 0.01$, and a discounted factor $\gamma = 0.8$. The algorithm is ran for $T = 15$ timesteps at each iteration. Moreover, stochastic actions are induced by randomizing the outcome of an action with a 5% probability.

9.2.1. Fetching task

In this scenario the state of the problem is represented as a 7-feature vector, where 3 components represent the distance of the robot end-effector to the target, 3 components encode the distance to an obstacle introduced in the scene, and the last component is the angle difference between the end-effector and world axis Z. We include such component to bias the agent in learning to fetch an object with a

preferable orientation. The reward function is a weighted sum of such components, and it is designed to promote states that are far from the obstacle, and close to the target position. Additionally, it penalizes states in which the end-effector does not point upwards, to simulate objects that have to be held with a preferred orientation (e.g. a glass full of water). The robot explores an action space composed by 13 actions: 6 translation actions to move the arm back and forth along the Cartesian axes, and 6 rotation actions to move the arm counter-/clockwise on the Roll, Pitch and Yaw angles. A `no-op` action is introduced to let the robot in its state. Figure 9.4a and Figure 9.4b illustrate obtained results by reporting the average cumulative reward and the number of explored states obtained during 10 iterations. In detail, reward values are averaged over 10 simulated fetching trials for each of the iterations and for each algorithm, the continuous lines represent average cumulative rewards while the line width their standard deviation. In the explored state plots, the width of the lines highlights the amount of states expanded during i with respect to the total number of states explored until $i-1$. While baseline algorithms perform worse in terms of obtained rewards (random-UCT, TD-search), only the vanilla-UCT shows results that are comparable to h-LoOP. However, the number of explored states of vanilla-UCT is significantly higher. Specifically, the naive implementation of UCT evaluates more than two times the number of states that h-LoOP ($\sim$55%). In fact, h-LoOP approximates the optimistic set of a HTN by evaluating only admissible actions that are expected to lead the search towards states with high reward. This is achieved by exploiting the action hierarchy updated at each iteration. To this extent, the results compare two different configurations of h-LoOP. The first, LoOP, is organized as a single layer structure, where the number of clusters within the layer is equal to the number of primitive actions, while the latter configuration, h-LoOP, is organized over 2 layers where the first layer also contains the set of primitive actions and, the second layer groups actions in 5 clusters. Again h-LoOP further reduces the number of explored states and confirms that a hierarchical evaluation of the search space improves the learning process. However, we do not notice a significant improvement between h-LoOP and LoOP. We associate cause of the comparable reward signals to the task complexity of the proposed scenario. In fact, differently from the "pick and delivery" task, the structure of "fetching" is not hierarchical and a hierarchical representation of Q-values may not be necessary. Still, we can observe that increasing the number of layers in the representation, even when that is not needed, does not damage the obtained performance and, still, slightly decreases the number of visited states. Additionally, it can be ran different evaluations to determine the optimal number of clusters within layers, however, for these scenarios, such a number is empirically chosen.

9.2.2. Pick and delivery task

Here the environment is represented as a 5x5 grid-world where the Pioneer has to collect an object at a random location and, carry it to an operator Figure 9.5. The scenario resembles the one addressed by the "taxi-agent" in [54], however a comparison with max-Q would not be proper since our reward is implemented to be *shaped* and not *sparse*; and we implement our approach in a robotic context where the reduced number of samples and iterations are limiting constraints. Here, the

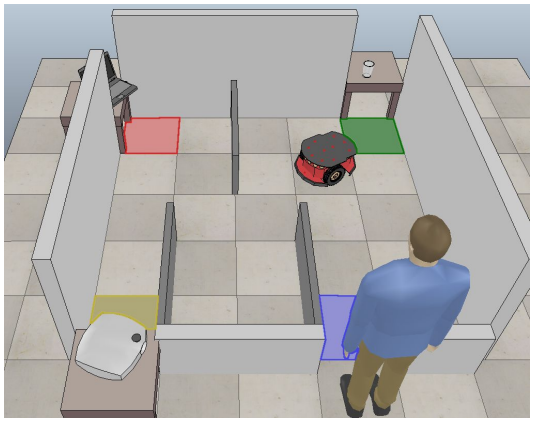
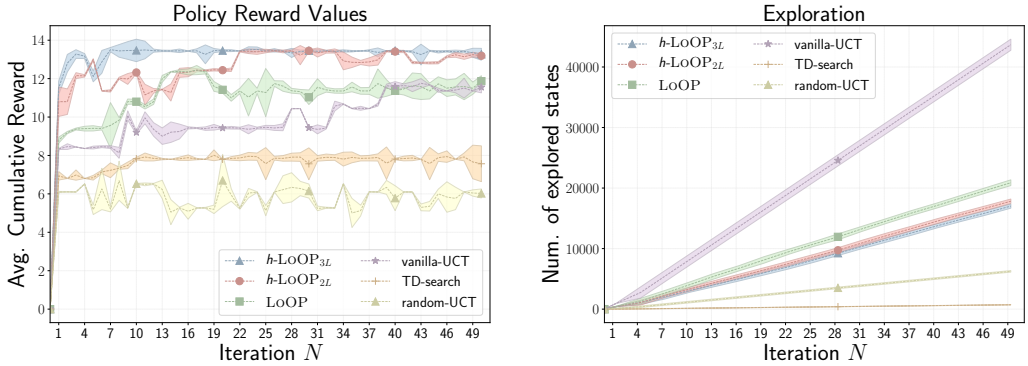


Figure 9.5. "Pick and delivery" scenario, the environment is composed by 4 working stations, the robot has to collect one item and delivery it to the operator (blue station).



(a) avg. cumulative reward **(b)** number of explored states

Figure 9.6. Average cumulative reward and number of explored states obtained by LoOP, h-LoOP$_{2L}$, h-LoOP$_{3L}$, TD-search, *random-UCT* and *vanilla-UCT* in 10 iterations in the Kuka fetching scenario. For each of them, the reward is averaged over 50 runs. The function approximator has been implemented with GMMs.

state is a 9-feature vector where the first two components represent the position of the robot, the following two encode the current target of the robot (either the object station or the delivery one), the fifth component indicates whether the object is picked, and the last four components indicate whether there is an obstacle in one of the four possible directions (e.g. wall). The action space of the agent in composed by 6 actions, four to move through cells, and two to pick and drop the object. Moreover, we assume that a robot is helped to collect and drop the objects by an external operator. The reward function is a weighted sum of two components encoding the distance of the robot to the target object and the distance of the object to its delivery station. This task represents a more complex scenario due to temporal constraints imposed by the status of the object, however, as in the previous scenario, a similar analysis of the results can be observed. Figure 9.6a and Figure 9.6b illustrate the average cumulative reward and the number of explored states obtained during 50 iterations. Vanilla-UCT has comparable reward values, but the number of explored

states is still significantly higher than each configuration of h-LoOP. The action hierarchy of h-LoOP, in fact, improves the overall performance showing the best results with a 3 layered structure. Particularly in these complex scenarios – where ordering constraints exist and the task can be decomposed – h-LoOP performs better and confirms that a multi-layered representation of affordance semantics improves the exploration of the search space.

9.3. Concluding Remarks

In this chapter, we extended the formulation of LoOP to hierarchical optimistic planning. The work described here aims at analyzing the correlation between κ-parameters and relations among κ-functions. In particular, whether it is possible to learn hierarchies of κ-functions. Hence, we centered our work around the question:

5. *We illustrated how* SK4R *can hierarchical define κ-functions to address reflection [131]. This lead us to investigate whether it is possible to learn hierarchies of κ-functions to support task decomposition by reusing ASs of primitive actions?*

This is an important feature of the SK4R representation that enables the κ-agent to reuse learned AS to represent compositions of actions and sub-tasks. Therefore, it is key to our goal of designing a practical and functional representation to support general purpose tasks, for an autonomous robot. To this end, we focused on the interaction of SK4R_P and the composition of κ-functions representing primitive robot actions, and, we introduced h-LoOP, an iterative learning algorithm of hierarchical value functions for optimistic planning. We described how (1) it determines hierarchies of affordance semantics within the SK4R representations by clustering actions at different levels of abstraction and how (2) it improves search space exploration in order to generate efficient policies. The experimental evaluation shows the efficacy of h-LoOP in enabling the agent to learn a good policy by evaluating a significant lower number of states. However, the reported results suggest that the benefits of applying the h-LoOP algorithm may vary in accordance with the application addressed. In fact, as we have seen in the fetching task scenario, if the structure of the problem is inherently not hierarchical, then the improvement of h-LoOP is not significantly higher with respect to LoOP. But, as expected the, even if the improvement is not remarkable, the algorithm does not damage the overall performance. Hence, such a result proves h-LoOP to have a better generalization capability. Conversely, in the case in which the problem addressed allows for a hierarchical formulation, the benefits demonstrated by h-LoOP are significant. As observed in the pick and place scenario. Still, the improvement carried out by different configurations of the actions hierarchies is not constant to the number of layers. This suggests that, in order to obtain a significant improvement, the configuration of h-LoOP has to be done carefully, and its design choices have to take into consideration the structure of the problem at hand.

However, h-LoOP can be used to solve several tasks in multiple domains. Moreover, as a future work strictly related to this chapter, we want to explore the

possibility of transferring hierarchical value functions among learning agents in order to take advantage of abstract actions and their semantics in different tasks.

Part III

Conclusion

Chapter 10

Conclusions

Realizing autonomous agents able properly to interpret the external world and operate in it is the ultimate goal of this thesis. In previous chapters, we motivated our work, conxtetualized it within the state-of-the-art and found different research aspects that, have demonstrated the validity of our contribution. Finally, this chapter concludes our dissertation by summarizing its content and highlighting its contributions. Throughout this chapter we will retrace the research problems addressed by this thesis, and outline the major findings along with. Then, a discussion is presented about further improvements and new directions for future work:

10.1. Summary and Contributions

The goal of this thesis is to investigate how a proper representation contributing spatial knowledge and knowledge about affordances can improve robot behaviors and aid action planning for general purpose tasks. Specifically, we centered our work around two questions that originated the research presented and motivate this thesis.

1. How can the robot represent and highlight important features of the environment to support actions execution?
2. How can the robot use spatial knowledge and knowledge about spatial affordances to design and learn goal-oriented behaviors to support general purpose tasks?

Hence, we find the motivations of our work in the need of deploying robotic agents able to (1) refer to the environment at different levels of abstractions – in order to interact with humans and plan actions for both local and global behaviors; (2) understand where and when an action supports task completion. The latter is key for an autonomous agent. It enables the robot to reason about the effects that its own actions have on the external world. In fact, the agent will no longer execute an action by means of a mere hand-crafted behavioral routine, rather, it will execute an action because it "knows" that the resulting state leads it one step closer to success in its task.

We studied how to enable effective behaviors by relying upon a proper representation of the operational environment, its current state and the task to be

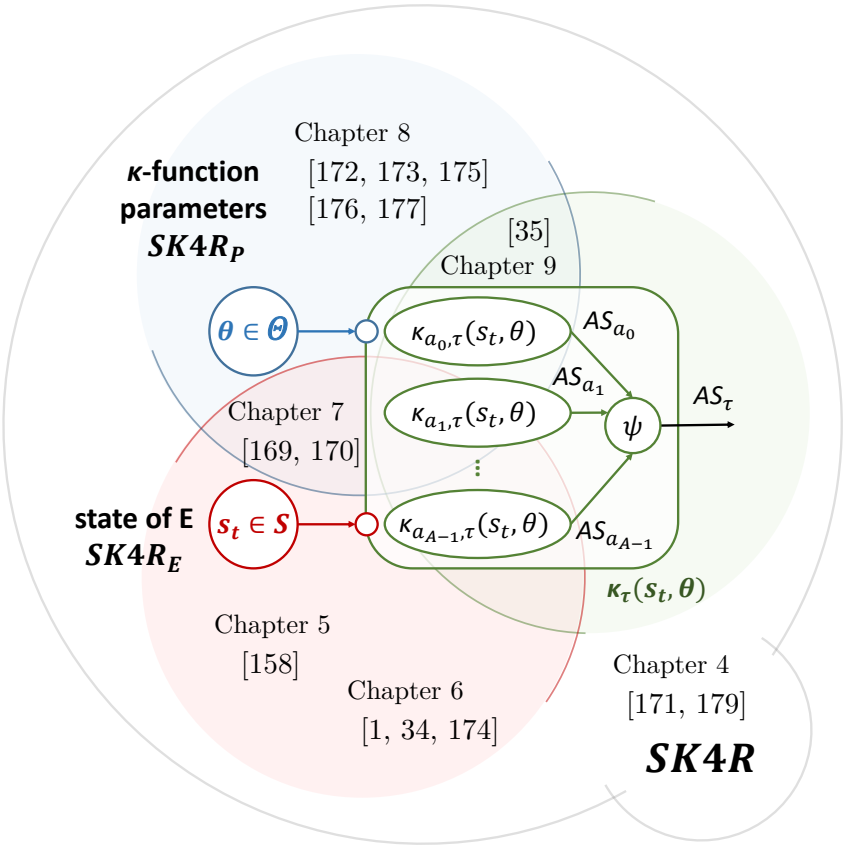


Figure 10.1. A representation for spatial knowledge for robots – SK4R. The working domain of each of the technical chapters is highlighted with different balloons and colors. References to chapters and related publications are added to provide a classification of the work of this thesis.

accomplished by the robot. To pursue such a goal, we have taken into consideration different research areas related to different robotic applications.

The first aspect addressed in this thesis has been the design of a spatial knowledge representation that could embrace a *functional* point of view about the environment and explicitly model robot actions. To this end, Chapter 4, introduces and describes the SK4R representation by highlighting its main features and by demonstrating how it can be exploited to encode affordance semantics. Figure 10.1 recalls the SK4R representation, and associates, the contribution of each chapter to a particular SK4R component. In each chapter, we investigated a different research topic that contributes to the formalization of SK4R, and thus, to dig into the manifold opportunities and challenges brought by the chosen representation:

- Chapter 5 is focused on the implementation of a spatial knowledge representation for an indoor service robot. The goal is to provide an implementation of the environmental module of SK4R (SK4R$_E$, red in Figure 10.1) able to generate a detailed representation of the state of the environment. To this end, we defined a hierarchical representation of spatial knowledge, whose layers

interact by means of a novel deep learning architecture, namely SPNs. The experimental evaluation shows that the proposed implementation of SK4R$_E$ can effectively label particular areas of the environment and infer missing knowledge by performing top-down inference. Additionally, such an implementation of SK4R$_E$ is able to encode the state of large environments and support action execution of indoor service robots. In particular, such an implementation enables long-term planning, and more reactive behaviors in the peripersonal domain of the robot.

- Chapter 6 further investigates the formalization of SK4R$_E$. In this case, however, we frame our contribution in the context of social robotics. We characterize the state of the environment through several social factors that may, or may not, enable acceptable behaviors when a human-robot social interaction has to be conducted. We evaluate different social factors in two user studies. The reported findings highlight which is the best social configuration that favors a more natural interaction. The goal of this chapter is to generate a set of guidelines for an autonomous robot that has to feature a social understanding of its operational environment.

- Chapter 7 investigates both how to define a domain-specific representation for multi-robot systems, and how to design κ-function parameters (i.e. $\boldsymbol{\theta}$) to support collective and individual behaviors. To this extent, this chapter implements both the environmental module SK4R$_E$, and the κ-function parameters module SK4R$_P$ (highlighted in red and in blue in Figure 10.1, respectively). The chapter contributes to a novel distributed coordination system able to adapt the team strategy to current configuration of the environment (e.g. poor network conditions). We introduced a context-aware coordination system that shares *events* in order to distributively reconstruct an approximate world model. Such a model is then used by each robot to deliberate the next action to execute. The experimental evaluation shows the benefits of a dynamic coordination strategy that, via κ-function parameters, can influence the behavior of the team and preserve a satisfying performance.

- Chapter 8 is dedicated to the SK4R$_P$ parameter module. This module is key to our representation as it exposes important parameters of the κ-functions that can be learned and exploited to influence affordance semantics of robot actions (in blue in Figure 10.1). Hence, SK4R$_P$ represents the main vehicle to shape and influence robot behaviors. To this end, the chapter introduces and describes LoOP, a novel iterative algorithm for optimistic planning with learned value functions. LoOP works as combination of MC planning and Q-learning, and makes robot planning practical in complex and different applications. The experimental evaluation shows that LoOP leads to a remarkable improvement in the computational load of the algorithm (with respect to other baseline algorithm) without loss in the performance. Such a feature makes our approach more practical in difficult robotics applications, where the lack of training examples is often a limiting condition.

- Chapter 9 extends the formulation of LoOP to a hierarchical formalization

of affordance semantics in order to allow for task decomposition. The study in this chapter is centered on two parts of the SK4R representation, namely the κ-function parameters module and the κ-functions (highlighted in blue and in green in Figure 10.1 respectively). To this end, we introduced h-LoOP, an iterative learning algorithm of hierarchical value functions for optimistic planning. h-LoOP is used to determine hierarchies among affordance semantics and guide the exploration at planning time. In fact, the experimental evaluation shows the effectiveness of h-LoOP in enabling the agent to learn a good policy by evaluating a significant lower number of states.

Finally, we recall that our dissertation investigates a spatial knowledge representation, by showing its benefits for several robotic applications and on different robotic platform. In each of the addressed scenarios, a proper formulation of spatial knowledge shows an improvement in the robot(s) performance and to support goal-oriented behaviors. In fact, the manifold experimental evaluations confirm that spatial knowledge can be used to better represent what the robot "knows" and how it should properly act in the external world.

In fact, a part from the important achievements of the applications in each chapter, our main contributions lie in the formalization of a new functional perspective in representing spatial knowledge and in the formalization of SK4R. We have demonstrated that having a representation explicitly modeling robot actions is key to adapt robot behaviors to complex environments. The paradigm proposed by the SK4R representation provides the robot with the possibility to structure and maintain (1) information about the environment and (2) knowledge about its own actions. These two aspects of SK4R enable to perform effective planning and provide the robot with the ability to highlight relevant aspects of the world with respect to what the it is capable of. This is extremely important, in fact, SK4R *only* represents aspects of the world relevant to characterize the task of the robot, and to support its actions. The contribution brought by SK4R constitutes a novel perspective in knowledge representation for an autonomous robot in unstructured environments. Such a perspective, in fact, is key to model what the robot can do in an environment, rather than, modeling the the environment as an entity separated from the robot itself.

10.2. Open Questions and Future Work

SK4R is a spatial knowledge representation ranging from world state representations to high-level action planning. To assess its effectiveness , we had the opportunity to traverse different research topics related to robotics, each of which can be further investigated more in depth. Hence, a considerable amount of open questions and new research directions might be suggested. To provide few examples:

1. In the contributions focusing on the formalization of the environment, key landmarks are selected by an expert before deployment. In contrast, it would be very interesting to let the robot discover important landmarks of the environment by itself. In general, once learned, the robot knowledge does not change. This leaves space to the study of an interesting aspect. How does

robot knowledge should continually be updated? Can the robot recognize knowledge which is actual meaningful to its scope? Humans always update their belief through continuous experience on daily basis. Is it possible to mimic such a skill? Is it beneficial in robotics?

2. Under which circumstances does spatial knowledge have to be updated? How can dynamic changes of the environment be addressed? Is it possible to select deprecated knowledge existing in SK4R$_E$, and erase it from the general state representation?

3. Specific to the multi-robot scenarios in Chapter 7, can contexts and events be discovered dynamically depending on the configuration of the robots environment? Can κ-parameters be learned in this scenario?

4. The discussion in Chapter 8 and Chapter 9 arise different new open questions. How policies learned to address a single task, can be combined to tackle compositions of them? To what extent is it possible to transfer policies among different agents? If yes, how does it change the affordance semantics with respect to the new platform?

Summarizing, by developing an end-to-end knowledge representation, this thesis faces interesting problems related to robotics and AI. Its goal is to determine important features of the operational environment in order to shape and support robots behaviors. In fact, the goal, pursued throughout this entire work, is to enable a robotic agent to exploit spatial knowledge in order to feature effective and practical capabilities. SK4R, indeed, can improve the performance of a robotic system in different application scenarios and in accomplishing various tasks.

After an evaluation of the research conducted within the context of this thesis, we can confirm that a proper representation of spatial knowledge and affordance semantics can enable robots to behave in complex environments and achieve a *human-compatible* understanding of the perceived world. This thesis develops an novel approach to model spatial knowledge and investigates different aspects where it can be effectively employed. Our objective was to validate the overall representation, and its individual components, by demonstrating their benefits and improvement to each considered robotic scenario.

But our dissertation achieves more. It defines new research questions that aim at better investigating new perspectives in representing spatial knowledge affordance semantics. Each of the future directions, defined by a more in-depth investigation of each SK4R components, requires a considerable effort that – along this thesis – we demonstrated to be worth investigating.

Bibliography

[1] Vanzo A., Riccio F., Sharf M., Mirabella V., Catarci T., and Nardi D. Who is willing to help robots? a user study on collaboration attitude. *journal paper under review*, Feb 2018. (Cited on pages 10, 58, 82, and 154.)

[2] Michal Abaffy, Tomáš Brázdil, Vojtěch Řehák, Branislav Bošanskỳ, Antonín Kučera, and Jan Krčál. Solving adversarial patrolling games with bounded error. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1617–1618. International Foundation for Autonomous Agents and Multiagent Systems, 2014. (Cited on page 106.)

[3] David Abel, D. Ellis Hershkowitz, Gabriel Barth-Maron, Stephen Brawner, Kevin O'Farrell, James MacGlashan, and Stefanie Tellex. Goal-based action priors. In *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015. (Cited on page 54.)

[4] F. Abtahi and M. R. Meybodi. Solving multi-agent markov decision processes using learning automata. In *2008 6th International Symposium on Intelligent Systems and Informatics*, pages 1–6, Sept 2008. doi: 10.1109/SISY.2008.4664909. (Cited on page 41.)

[5] A. Agostini and E. Celaya. Reinforcement learning with a gaussian mixture model. In *The 2010 International Joint Conference on Neural Networks*, pages 1–8, July 2010. doi: 10.1109/IJCNN.2010.5596306. (Cited on pages 118, 140, and 143.)

[6] Ankit Anand, Aditya Grover, Mausam Mausam, and Parag Singla. Asap-uct: Abstraction of state-action pairs in uct. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1509–1515. AAAI Press, 2015. ISBN 978-1-57735-738-4. URL `http://dl.acm.org/citation.cfm?id=2832415.2832459`. (Cited on page 138.)

[7] D. Anguelov, D. Koller, E. Parker, and S. Thrun. Detecting and modeling doors with mobile robots. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 3777–3784 Vol.4, April 2004. doi: 10.1109/ROBOT.2004.1308857. (Cited on page 3.)

[8] B. D. Argall, E. L. Sauser, and A. G. Billard. Tactile guidance for policy refinement and reuse. In *2010 IEEE 9th International Conference on Development and Learning*, pages 7–12, Aug 2010. doi: 10.1109/DEVLRN.2010.5578872. (Cited on pages 39 and 42.)

[9] 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 29.)

[10] J. Aulinas, Y.R. Petillot, J. Salvi, and X. Lladó. The slam problem: a survey. *Frontiers in Artificial Intelligence and Applications*, 184:363–371, Oct. 22-24 2008. (Cited on page 36.)

[11] A. Aydemir, A. Pronobis, M. Göbelbecker, and P. Jensfelt. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, Aug 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2256686. (Cited on pages 6 and 61.)

[12] E. Babaians, N. Khazaee Korghond, A. Ahmadi, M. Karimi, and S. S. Ghidary. Skeleton and visual tracking fusion for human following task of service robots. In *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pages 761–766, Oct 2015. doi: 10.1109/ICRoM.2015.7367878. (Cited on page 54.)

[13] Usman Babawuro and Zou Beiji. Knowledge representation: a general survey and techniques for sound knowledge based systems. *Int J Intell Inf Process*, 2(4):16–22, 2011. (Cited on page 15.)

[14] 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 39.)

[15] 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 23 and 37.)

[16] M. R. Batista, A. H. M. Pinto, and R. A. F. Romero. Addressing escorting by behavior combining using multiple differential drive robots. In *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, pages 187–191, Oct 2015. doi: 10.1109/LARS-SBR.2015.55. (Cited on page 54.)

[17] Daniel S. Bernstein, Christopher Amato, Eric A. Hansen, and Shlomo Zilberstein. Policy iteration for decentralized control of markov decision processes. *CoRR*, abs/1401.3460, 2014. URL `http://arxiv.org/abs/1401.3460`. (Cited on page 41.)

[18] 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 23.)

[19] Domenico D. Bloisi, D. Nardi, F. Riccio, and F. Trapani. *Context in Robotics and Information Fusion*, pages 675–699. Springer International Publishing, 2016. ISBN 978-3-319-28971-7. doi: 10.1007/978-3-319-28971-7_25. (Cited on page 9.)

[20] T. M. Bonanni, B. Della Corte, and G. Grisetti. 3-d map merging on pose graphs. *IEEE Robotics and Automation Letters*, 2(2):1031–1038, April 2017. doi: 10.1109/LRA.2017.2655139. (Cited on page 36.)

[21] Alejandro Bordallo, Fabio Previtali, Nantas Nardelli, and Subramanian Ramamoorthy. Counterfactual reasoning about intent for interactive navigation in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2943–2950, 2015. doi: 978-1-4799-9993-4. (Cited on pages 2 and 3.)

[22] D. Boud, R. Keogh, and D. Walker. *Reflection: Turning Experience Into Learning*. Kogan Page, 1985. ISBN 9780850388640. (Cited on pages 6 and 50.)

[23] Michael Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 89–94. Morgan Kaufmann, 2000. (Cited on page 121.)

[24] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool. Robust tracking-by-detection using a detector confidence particle filter. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1515–1522, Sept 2009. doi: 10.1109/ICCV.2009.5459278. (Cited on page 3.)

[25] E. Broadbent, R. Stafford, and B. MacDonald. Acceptance of healthcare robots for the older population: Review and future directions. *International Journal of Social Robotics*, 1(4):319, Oct 2009. ISSN 1875-4805. doi: 10.1007/s12369-009-0030-6. URL `https://doi.org/10.1007/s12369-009-0030-6`. (Cited on page 1.)

[26] 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 29, 38, 112, and 140.)

[27] Anders Glent Buch, Dirk Kraft, Joni-Kristian Kämŕäinen, Henrik Gordon Petersen, and Norbert Krüger. *Pose Estimation using Local Structure-Specific Shape and Appearance Context*, pages 2080–2087. IEEE, United States, 2013. ISBN 978-1-4673-5641-1. doi: 10.1109/ICRA.2013.6630856. (Cited on page 18.)

[28] Lucian Busoniu and Remi Munos. Optimistic Planning for Markov Decision Processes. In *15th International Conference on Artificial Intelligence and Statistics, AISTATS-12*, volume 22 of *Journal of Machine Learning Research: Workshop and Conference Proceedings*, pages 182–189, La Palma, Canary Islands, Spain, April 2012. URL `https://hal.archives-ouvertes.fr/hal-00756736`. (Cited on page 29.)

[29] 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 112.)

[30] Y. Uny Cao, Alex S. Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, Mar 1997. ISSN 1573-7527. doi: 10.1023/A:1008855018923. URL `https://doi.org/10.1023/A:1008855018923`. (Cited on page 40.)

[31] J. Capitan, M. T. J. Spaan, L. Merino, and A. Ollero. Decentralized multi-robot cooperation with auctioned pomdps. In *2012 IEEE International Conference on Robotics and Automation*, pages 3323–3328, May 2012. doi: 10.1109/ICRA.2012.6224917. (Cited on pages 41 and 96.)

[32] R. Capobianco, Guglielmo Gemignani, Domenico Bloisi, D. Nardi, and L. 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 page 37.)

[33] R. Capobianco, J. Serafin, J. Dichtl, G. Grisetti, L. Iocchi, and D. 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 ix and 22.)

[34] R. Capobianco, Guglielmo Gemignani, L. Iocchi, D. Nardi, F. Riccio, and A. Vanzo. Contexts for symbiotic autonomy: Semantic mapping, task teaching and social robotics. In Jeffrey O. Kephart, Stephanie Rosenthal, Manuela M. Veloso, and Alex Rudnicky, editors, *Symbiotic Cognitive Systems, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 13, 2016.*, volume WS-16-14 of *AAAI Workshops*, pages 733–736, Phoenix, Arizona, USA, February 2016. AAAI Press. (Cited on pages 10, 58, and 154.)

[35] R. Capobianco, F. Riccio, and D. Nardi. Hi-val: Iterative learning of hierarchical value functions for policy generation. In *Proceedings of the 2018 fifteenth International Conference on Intelligent Autonomous Systems IAS-15*, pages –, Feb 2018. (Cited on pages 8, 11, 58, 112, 138, and 154.)

[36] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty: discrete bayesian models for mobile-robot navigation. In *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 2, pages 963–972 vol.2, Nov 1996. doi: 10.1109/IROS.1996.571080. (Cited on page 2.)

[37] Girish Chowdhary, Miao Liu, Robert Grande, Thomas Walsh, Jonathan How, and Lawrence Carin. Off-policy reinforcement learning with gaussian processes. *IEEE/CAA Journal of Automatica Sinica*, 1(3):227–238, 2014. (Cited on page 118.)

[38] Michael Jae-Yoon Chung, Andrzej Pronobis, Maya Cakmak, Dieter Fox, and Rajesh P. N. Rao. Autonomous question answering with mobile robots in human-populated environments. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, October 2016. doi: 10.1109/IROS.2016.7759146. URL `http://www.pronobis.pro/publications/chung2016iros`. (Cited on pages 22, 35, 36, and 68.)

[39] Aaron St Clair, Carl Saldanha, Adrian Boteanu, and Sonia Chernova. Interactive hierarchical task learning via crowdsourcing for robot adaptability. In *Refereed workshop Planning for Human-Robot Interaction: Shared Autonomy and Collaborative Robotics at Robotics: Science and Systems, Ann Arbor, Michigan. RSS*, 2016. (Cited on page 138.)

[40] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 746–752, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL `http://dl.acm.org/citation.cfm?id=295240.295800`. (Cited on page 121.)

[41] S. Coradeschi and A. Saffiotti. Symbiotic robotic systems: Humans, robots, and smart environments. *IEEE Intelligent Systems*, 21(3):82–84, 2006. (Cited on page 82.)

[42] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on page 38.)

[43] A. Corrêa. Distributed team formation in urban disaster environments. In *2014 IEEE Symposium on Intelligent Agents (IA)*, pages 57–64, Dec 2014. doi: 10.1109/IA.2014.7009459. (Cited on page 41.)

[44] G. Costante, T. A. Ciarfuglia, P. Valigi, and E. Ricci. A transfer learning approach for multi-cue semantic place recognition. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2122–2129, Nov 2013. doi: 10.1109/IROS.2013.6696653. (Cited on page 18.)

[45] Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation. *AI Magazine*, 14(1), 1993. (Cited on page 5.)

[46] G De Giacomo, L Iocchi, D Nardi, and R Rosati. A theory and implementation of cognitive mobile robots. *Journal of Logic and Computation*, 9(5):759–785, 1999. doi: 10.1093/logcom/9.5.759. URL `http://dx.doi.org/10.1093/logcom/9.5.759`. (Cited on pages 1 and 6.)

[47] Maartje M.A. de Graaf and Somaya Ben Allouch. Expectation setting and personality attribution in hri. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 144–145, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2658-2. doi: 10.1145/2559636.2559796. URL `http://doi.acm.org/10.1145/2559636.2559796`. (Cited on page 93.)

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. doi: 10.2307/2984875. (Cited on page 119.)

[49] S. Dhelim, H. Ning, and T. Zhu. Stlf: Spatial-temporal-logical knowledge representation and object mapping framework. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 001550–001554, Oct 2016. doi: 10.1109/SMC.2016.7844459. (Cited on page 23.)

[50] Maurilio Di Cicco, Luca Iocchi, and Giorgio Grisetti. Nonparametric calibration for depth sensors. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, pages 923–935. Springer International Publishing, 2016. ISBN 978-3-319-08338-4. (Cited on page 36.)

[51] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, July 2006. ISSN 0018-9219. doi: 10.1109/JPROC.2006.876939. (Cited on page 41.)

[52] M. Bernrdine Dias and Anthony (Tony) Stentz. Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. Technical report, In Technical report, CMU-RI, 2003. (Cited on page 98.)

[53] G. Diego and T. 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.)

[54] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research (JAIR)*, 13:227–303, 2000. (Cited on pages 137, 139, 145, and 146.)

[55] Christian Dondrup, Marc Hanheide, Nicola Bellotto, et al. A probabilistic model of human-robot spatial interaction using a qualitative trajectory calculus. *AAAI Spring Symposium*, 2014. (Cited on pages 38 and 48.)

[56] 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 page 6.)

[57] Gregory Dudek, Michael R. M. Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, Dec 1996. ISSN 1573-7527. doi: 10.1007/BF00240651. URL `https://doi.org/10.1007/BF00240651`. (Cited on page 40.)

[58] Christoph Engel. Dictator games: a meta study. *Experimental Economics*, 14:583–610, 2011. (Cited on page 92.)

[59] 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 37 and 48.)

[60] Kutluhan Erol, James Hendler, and Dana S. Nau. Htn planning: Complexity and expressivity. In *the Twelfth National Conference on Artificial Intelligence (Vol. 2)*, AAAI'94, pages 1123–1128, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. ISBN 0-262-61102-3. URL `http://dl.acm.org/citation.cfm?id=199480.199459`. (Cited on pages 8, 43, 51, 138, and 141.)

[61] A. Fabbri, F. Armetta, É. Duchêne, and S. Hassas. Knowledge complement for monte carlo tree search: An application to combinatorial games. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, pages 997–1003, Nov 2014. doi: 10.1109/ICTAI.2014.151. (Cited on pages 39 and 43.)

[62] A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028, Oct 2004. ISSN 1083-4419. doi: 10.1109/TSMCB.2004.832155. (Cited on pages 40 and 96.)

[63] Kerstin Fischer, Stephen Yang, Brian Mok, Rohan Maheshwari, David Sirkin, and Wendy Ju. Initiating interactions and negotiating approach: A robotic trash can in the field. In *AAAI Symposium on Turn-taking and Coordination in Human-Machine Interaction*, pages 10–16, 2015. (Cited on page 82.)

[64] Michalis Foukarakis, Asterios Leonidis, Margherita Antona, and Constantine Stephanidis. Combining finite state machine and decision-making tools for adaptable robot behavior. In Constantine Stephanidis and Margherita Antona, editors, *Universal Access in Human-Computer Interaction. Aging and Assistive Environments*, pages 625–635, Cham, 2014. Springer International Publishing. ISBN 978-3-319-07446-7. (Cited on page 24.)

[65] F. Fraundorfer, C. Engels, and D. Nister. Topological mapping, localization and navigation using image collections. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3877, Oct 2007. doi: 10.1109/IROS.2007.4399123. (Cited on page 36.)

[66] 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 page 37.)

[67] Alberto Garcia-Garcia, Sergio Orts, Sergiu Oprea, Victor Villena-Martinez, and José García Rodríguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, 2017. (Cited on page 38.)

[68] Sylvain Gelly and David Silver. Combining online and offline knowledge in uct. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007. (Cited on page 136.)

[69] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856 – 1875, 2011. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2011.03.007. URL `http://www.sciencedirect.com/science/article/pii/S000437021100052X`. (Cited on pages 39 and 42.)

[70] Guglielmo Gemignani, Roberto Capobianco, Emanuele Bastianelli, Domenico Daniele Bloisi, Luca Iocchi, and Daniele Nardi. Living with robots: Interactive environmental knowledge acquisition. *Robotics and Autonomous Systems*, 78:1 – 16, 2016. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2015.11.001. URL `http://www.sciencedirect.com/science/article/pii/S0921889015002468`. (Cited on pages 19, 23, 35, 37, 42, and 48.)

[71] Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 3239–3247, USA, 2012. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999325.2999496`. (Cited on page 71.)

[72] Christopher Geyer. Active target search from uavs in urban environments. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2366–2371. IEEE, 2008. (Cited on page 106.)

[73] James Jerome Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979. ISBN 0-395-27049-9. Includes indexes. (Cited on pages 37 and 48.)

[74] Robert Glaser. Education and thinking: The role of knowledge. *American psychologist*, 39(2):93, 1984. (Cited on page 6.)

[75] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. (Cited on page 121.)

[76] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, Feb 2007. ISSN 1552-3098. doi: 10.1109/TRO.2006.889486. (Cited on pages 20 and 67.)

[77] Dylan Hadfield-Menell, Anca D. Dragan, Pieter Abbeel, and Stuart J. Russell. Cooperative inverse reinforcement learning. *CoRR*, abs/1606.03137, 2016. URL `http://arxiv.org/abs/1606.03137`. (Cited on page 41.)

[78] E. T. Hall. *The Hidden Dimension: Man's Use of Space in Public and Private*. The Bodley Head Ltd, 1966. (Cited on page 86.)

[79] Marc Hanheide, Moritz Göbelbecker, Graham S. Horn, A. Pronobis, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, Hendrik Zender, Geert-Jan Kruijff, Nick Hawes, and Jeremy L. Wyatt. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 2016. doi: 10.1016/j.artint.2015.08.008. (Cited on pages 5, 6, 19, 23, 35, 61, and 66.)

[80] M. Hassan, D. Liu, and G. Paul. Modeling and stochastic optimization of complete coverage under uncertainties in multi-robot base placements. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2978–2984, Oct 2016. doi: 10.1109/IROS.2016.7759461. (Cited on pages 15 and 41.)

[81] Nick Hawes, Hendrik Zender, Kristoffer Sjöö, Michael Brenner, Geert-Jan Kruijff, and Patric Jensfelt. Planning and acting with an integrated sense of space. In *Proc. of the International Workshop on Hybrid Control of Autonomous Systems*, 7 2009. (Cited on page 6.)

[82] D. Held, J. Levinson, and S. Thrun. A probabilistic framework for car detection in images using context and scale. In *2012 IEEE International Conference on Robotics and Automation*, pages 1628–1634, May 2012. doi: 10.1109/ICRA.2012.6224722. (Cited on page 19.)

[83] N. Hirose, R. Tajima, and K. Sukigara. Personal robot assisting transportation to support active human life: Human-following method based on model predictive control for adjacency without collision. In *2015 IEEE International Conference on Mechatronics (ICM)*, pages 76–81, March 2015. doi: 10.1109/ICMECH.2015.7083951. (Cited on page 54.)

[84] M. D. Hoang, S. S. Yun, and J. S. Choi. The reliable recovery mechanism for person-following robot in case of missing target. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 800–803, June 2017. doi: 10.1109/URAI.2017.7992828. (Cited on page 54.)

[85] G. Hollinger and S. Singh. Multi-robot coordination with periodic connectivity. In *2010 IEEE International Conference on Robotics and Automation*, pages 4457–4462, May 2010. doi: 10.1109/ROBOT.2010.5509175. (Cited on page 104.)

[86] Geoffrey Hollinger, Sanjiv Singh, Joseph Djugash, and Athanasios Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201–219, 2009. (Cited on page 106.)

[87] Geoffrey Hollinger, Srinivas Yerramalli, Sanjiv Singh, Urbashi Mitra, Gaurav S Sukhatme, et al. Distributed coordination and data fusion for underwater search. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 349–355. IEEE, 2011. (Cited on page 96.)

[88] Nicholas P Holmes and Charles Spence. The body schema and multisensory representation(s) of peripersonal space. *Cognitive processing*, 5(2), 2004. (Cited on page 65.)

[89] Jesse Hostetler, Alan Fern, and Thomas G. Dietterich. Sample-based tree search with fixed and adaptive state abstractions. *J. Artif. Intell. Res.*, 60:717–777, 2017. doi: 10.1613/jair.5483. URL `https://doi.org/10.1613/jair.5483`. (Cited on page 138.)

[90] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006. doi: 10.1177/0278364906072250. URL `https://doi.org/10.1177/0278364906072250`. (Cited on pages 41 and 96.)

[91] D. Jain, L. Mosenlechner, and M. Beetz. Equipping robot control programs with first-order probabilistic reasoning capabilities. In *2009 IEEE International Conference on Robotics and Automation*, pages 3626–3631, May 2009. doi: 10.1109/ROBOT.2009.5152676. (Cited on page 5.)

[92] Sang-Wook Jeon, Doo-Sung Ahn, Hyo-Jeong Bae, and Chang-Woo Hong. Object contour following task based on integrated information of vision and force sensor. In *2007 International Conference on Control, Automation and Systems*, pages 1040–1045, Oct 2007. doi: 10.1109/ICCAS.2007.4407051. (Cited on page 2.)

[93] A. Jevtić, G. Doisy, Y. Parmet, and Y. Edan. Comparison of interaction modalities for mobile indoor robot guidance: Direct physical interaction, person following, and pointing control. *IEEE Transactions on Human-Machine Systems*, 45(6):653–663, Dec 2015. ISSN 2168-2291. doi: 10.1109/THMS.2015.2461683. (Cited on pages 6 and 54.)

[94] Morimoto Jun and Doya Kenji. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems*, 36(1):37 – 51, 2001. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/S0921-8890(01)00113-0. (Cited on pages 39 and 138.)

[95] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. (Cited on page 39.)

[96] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, Dec 2008. ISSN 1552-3098. doi: 10.1109/TRO.2008.2006706. (Cited on page 36.)

[97] 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 pages 37 and 42.)

[98] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt*. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483, May 2011. doi: 10.1109/ICRA.2011.5980479. (Cited on page 38.)

[99] Sertac Karapinar and Sanem Sariel. Cognitive robots learning failure contexts through real-world experimentation. *Auton. Robots*, 39(4):469–485, December 2015. ISSN 0929-5593. doi: 10.1007/s10514-015-9471-y. URL `http://dx.doi.org/10.1007/s10514-015-9471-y`. (Cited on page 18.)

[100] Dov Katz, Arun Venkatraman , Moslem Kazemi, J. Andrew (Drew) Bagnell, and Anthony (Tony) Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. In *Robotics: Science and Systems Conference*, June 2013. (Cited on page 38.)

[101] Harmish Khambhaita and Rachid Alami. Assessing the Social Criteria for Human-Robot Collaborative Navigation: A Comparison of Human-Aware Navigation Planners. In *Proc. IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, page 6p., Lisbonne, Portugal, August 2017. URL `https://hal.laas.fr/hal-01568841`. (Cited on page 81.)

[102] Beomjoon Kim and Joelle Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8 (1):51–66, 2016. ISSN 1875-4805". doi: 10.1007/s12369-015-0310-2. (Cited on page 39.)

[103] 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 page 48.)

[104] Kheng Lee Koay, Dag Sverre Syrdal, Mohammadreza Ashgari-Oskoei, Michael L. Walters, and Kerstin Dautenhahn. Social roles and baseline proxemic preferences for a domestic service robot. *International Journal of Social Robotics*, 6:469–488, 2014. (Cited on page 86.)

[105] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009. (Cited on page 39.)

[106] 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 39.)

[107] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. *Machine learning: ECML 2006*, pages 282–293, 2006. (Cited on pages 112 and 117.)

[108] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *2004 IEEE International Conference on Robotics and Automation*, volume 3, pages 2619–2624 Vol.3, April 2004. doi: 10.1109/ROBOT.2004.1307456. (Cited on page 39.)

[109] 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 39 and 138.)

[110] Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 244–252, USA, 2011. Curran Associates Inc. ISBN 978-1-61839-599-3. URL `http://dl.acm.org/citation.cfm?id=2986459.2986487`. (Cited on page 38.)

[111] 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 page 48.)

[112] Dimitrios Kosmopoulos, Ilias Maglogiannis, and Fillia Makedon. Robust offline topological map estimation using visual loop closures. In *Proceedings of the 6th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '13, pages 29:1–29:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1973-7. doi: 10.1145/2504335.2504366. URL `http://doi.acm.org/10.1145/2504335.2504366`. (Cited on page 22.)

[113] 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: https://doi.org/10.1016/j.robot.2014.12.006. (Cited on page 36.)

[114] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999134.2999257`. (Cited on page 29.)

[115] 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 37 and 48.)

[116] 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 5.)

[117] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, May 2011. doi: 10.1109/ICRA.2011.5979949. (Cited on page 36.)

[118] John E. Laird and Shiwali Mohan. A case study of knowledge integration across multiple memories in soar. *Biologically Inspired Cognitive Architectures*, 8:93 – 99, 2014. ISSN 2212-683X. doi: https://doi.org/10.1016/j.bica.2014.03.006. (Cited on page 18.)

[119] 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 112.)

[120] 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 pages 40, 61, and 112.)

[121] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. (Cited on page 112.)

[122] Ming Liu, Francis Colas, Luc Oth, and Roland Siegwart. Incremental topological segmentation for semi-structured environments using discretized gvg. *Autonomous Robots*, 38(2):143–160, Feb 2015. ISSN 1573-7527. doi: 10.1007/s10514-014-9398-8. URL `https://doi.org/10.1007/s10514-014-9398-8`. (Cited on page 36.)

[123] Z. Liu, E. Blasch, Z. Xue, J. Zhao, R. Laganiere, and W. Wu. Objective assessment of multiresolution image fusion algorithms for context enhancement in night vision: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):94–109, Jan 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.109. (Cited on page 18.)

[124] Z. Liu, D. Chen, K. M. Wurm, and G. von Wichert. Using rule-based context knowledge to model table-top scenes. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2646–2651, May 2014. doi: 10.1109/ICRA.2014.6907238. (Cited on page 18.)

[125] 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 pages 37 and 38.)

[126] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, Oct 1997. ISSN 1573-7527. doi: 10.1023/A:1008854305733. URL `https://doi.org/10.1023/A:1008854305733`. (Cited on page 36.)

[127] 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 page 37.)

[128] L. Luo, N. Chakraborty, and K. Sycara. Competitive analysis of repeated greedy auction algorithm for online multi-robot task assignment. In *2012 IEEE International Conference on Robotics and Automation*, pages 4792–4799, May 2012. doi: 10.1109/ICRA.2012.6225195. (Cited on pages 41 and 96.)

[129] Patrick MacAlpine, Francisco Barrera, and Peter Stone. Positioning to win: A dynamic role assignment and formation positioning system. In Xiaoping Chen, Peter Stone, Luis Enrique Sucar, and Tijn van der Zant, editors, *RoboCup 2012: Robot Soccer World Cup XVI*, pages 190–201, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39250-4. (Cited on pages 41 and 96.)

[130] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*, pages 281–297. University of California Press, Berkeley, CA, USA, 1967. (Cited on page 119.)

[131] Pattie Maes and D. Nardi. *Meta-Level Architectures and Reflection*. Elsevier Science Inc., New York, NY, USA, 1988. ISBN 0444703438. (Cited on pages 6, 8, 50, 59, and 148.)

[132] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *Proc. of ICRA*, 2010. (Cited on pages 3, 5, 6, and 69.)

[133] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita. Adapting robot behavior for human–robot interaction. *IEEE Transactions on Robotics*, 24(4):911–916, Aug 2008. ISSN 1552-3098. doi: 10.1109/TRO.2008.926867. (Cited on pages 2 and 88.)

[134] 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. URL `http://dx.doi.org/10.1038/nature14236`. (Cited on pages x, 8, 29, 31, 32, 39, 112, 120, 121, and 123.)

[135] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016. (Cited on pages 40, 43, 112, 120, and 136.)

[136] O. M. Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using adaboost. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1730–1735, April 2005. doi: 10.1109/ROBOT.2005.1570363. (Cited on page 75.)

[137] Jonathan Mumm and Bilge Mutlu. Human-robot proxemics: Physical and psychological distancing in human-robot interaction. In *Proceedings of the 6th International Conference on Human-robot Interaction*, HRI '11, pages 331–338. ACM, 2011. (Cited on pages 88 and 92.)

[138] Krishna Kumar Narayanan, Luis-Felipe Posada, Frank Hoffmann, and Torsten Bertram. Situated learning of visual robot behaviors. In Sabina Jeschke, Honghai Liu, and Daniel Schilberg, editors, *Intelligent Robotics and Applications*, pages 172–182, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25486-4. (Cited on page 17.)

[139] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950. ISSN 0027-8424. doi: 10.1073/pnas.36.1.48. URL `http://www.pnas.org/content/36/1/48`. (Cited on page 27.)

[140] Q. V. Nguyen, F. Colas, E. Vincent, and F. Charpillet. Long-term robot motion planning for active sound source localization with monte carlo tree search. In *2017 Hands-free Speech Communications and Microphone Arrays (HSCMA)*, pages 61–65, March 2017. doi: 10.1109/HSCMA.2017.7895562. (Cited on pages 38 and 39.)

[141] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, pages 189–196, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-2883-8. doi: 10.1145/2696454.2696455. URL `http://doi.acm.org/10.1145/2696454.2696455`. (Cited on pages 39 and 82.)

[142] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915 – 926, 2008. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.08.001. URL `http://www.sciencedirect.com/science/article/pii/S0921889008001127`. Semantic Knowledge in Robotics. (Cited on page 37.)

[143] Dominik Off and Jianwei Zhang. Continual htn robot task planning in open-ended domains: A case study. *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011. (Cited on page 66.)

[144] Steven Okamoto, Nathan Brooks, Sean Owens, Katia Sycara, and Paul Scerri. Allocating spatially distributed tasks in large, dynamic robot teams. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pages 1245–1246, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-9826571-7-X, 978-0-9826571-7-1. URL `http://dl.acm.org/citation.cfm?id=2034396.2034508`. (Cited on pages 41, 42, and 96.)

[145] S. Ontañón. Informed monte carlo tree search for real-time strategy games. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, Sept 2016. doi: 10.1109/CIG.2016.7860394. (Cited on page 39.)

[146] 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 page 48.)

[147] Lynne E. Parker, George Bekey, and Jacob Barhen. *Current State of the Art in Distributed Autonomous Mobile Robotics*, pages 3–12. Springer Japan, Tokyo, 2000. ISBN 978-4-431-67919-6. doi: 10.1007/978-4-431-67919-6_1. URL `https://doi.org/10.1007/978-4-431-67919-6_1`. (Cited on page 40.)

[148] Rohan Paul, Jacob Arkin, Nicholas Roy, and Thomas M. Howard. Grounding abstract spatial concepts for language interaction with robots. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4929–4933, 2017. doi: 10.24963/ijcai.2017/696. URL `https://doi.org/10.24963/ijcai.2017/696`. (Cited on page 1.)

[149] Mikkel Rath Pedersen, Lazaros Nalpantidis, Rasmus Skovgaard Andersen, Casper Schou, Simon Bøgh, Volker Krüger, and Ole Madsen. Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, 37:282 – 291, 2016. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2015.04.002. URL `http://www.sciencedirect.com/science/article/pii/S0736584515000575`. (Cited on page 1.)

[150] Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, and Pedro Domingos. On theoretical properties of sum-product networks. *Journal of Machine Learning Research*, 38:744–752, 2015. ISSN 1532-4435. (Cited on page 71.)

[151] Svetlin Penkov, Alejandro Bordallo, and Subramanian Ramamoorthy. Physical symbol grounding and instance learning through demonstration and eye tracking. In *Robotics and Automation, 2017 IEEE International Conference on*, Singapore, June 2017. (Cited on page 112.)

[152] Tiago Pereira, António Paulo Moreira, and Manuela Veloso. Coordination for multi-robot exploration using topological maps. In António Paulo Moreira, Aníbal Matos, and Germano Veiga, editors, *CONTROLO'2014 – Proceedings of the 11th Portuguese Conference on Automatic Control*, pages 515–524, Cham, 2015. Springer International Publishing. ISBN 978-3-319-10380-8. (Cited on page 41.)

[153] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690, Nov 2011. doi: 10.1109/ICCVW.2011.6130310. (Cited on pages 71 and 72.)

[154] Erwin Prassler, Rainer Bischoff, Wolfram Burgard, Robert Haschke, Martin H. Gele, Gisbert Lawitzky, Bernhard Nebel, Paul Pl Ger, Ulrich Reiser, Marius Z. Llner, Martin Hagele, Paul Ploger, and Marius Zollner. *Towards Service Robots for Everyday Environments: Recent Advances in Designing Service Robots for Complex Tasks in Everyday Environments*. Springer Publishing Company, Incorporated, 2016. ISBN 3662508583, 9783662508589. (Cited on page 1.)

[155] A. 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 ix, 1, 23, 37, 42, and 74.)

[156] A. Pronobis, O. Martínez Mozos, B. Caputo, and P. Jensfelt. Multi-modal semantic place classification. *Int. J. Rob. Res.*, 29(2-3):298–320, February 2010. ISSN 0278-3649. doi: 10.1177/0278364909356483. URL `http://dx.doi.org/10.1177/0278364909356483`. (Cited on page 75.)

[157] A. Pronobis, Kristoffer Sjöö, Alper Aydemir, Adrian N. Bishop, and Patric Jensfelt. Representing spatial knowledge in mobile cognitive systems. In *Proc. of the International Conference on Intelligent Autonomous Systems (IAS-11)*, August 2010. (Cited on pages 5 and 66.)

[158] A. Pronobis, F. Riccio, and R. P. N. Rao. Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments. In *ICAPS 2017 Workshop on Planning and Robotics*, Pittsburgh, PA, USA, jun 2017. (Cited on pages 2, 10, 19, 21, 22, 35, 36, 58, and 154.)

[159] A. Pronobis, F. Riccio, and R. P. N. Rao. Deep Spatial Affordance Hierarchy: Spatial knowledge representation for planning in large-scale environments. In *RSS 2017 Workshop on Spatial-Semantic Representations in Robotics*, Boston, MA, USA, jul 2017. (Cited on pages 10, 71, and 72.)

[160] Andrzej Pronobis and Rajesh P. N. Rao. Learning deep generative spatial models for mobile robots. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, September 2017. doi: 10.1109/IROS.2017.8202235. (Cited on pages 64 and 72.)

[161] Andrzej Pronobis, Patric Jensfelt, Kristoffer Sjöö, Hendrik Zender, Geert-Jan M. Kruijff, Oscar M. Mozos, and Wolfram Burgard. Semantic modelling of space. In Henrik I. Christensen, Geert-Jan M. Kruijff, and Jeremy L. Wyatt, editors, *Cognitive Systems*, volume 8 of *Cognitive Systems Monographs*, pages 165–221. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-11694-0. doi: 10.1007/978-3-642-11694-0_5. (Cited on pages ix, 21, and 23.)

[162] Scott Proper and Prasad Tadepalli. Solving multiagent assignment markov decision processes. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 681–688, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9817381-6-1. (Cited on page 41.)

[163] 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 15.)

[164] O. A. I. Ramírez, H. Khambhaita, R. Chatila, M. Chetouani, and R. Alami. Robots learning how and where to approach people. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 347–353, Aug 2016. doi: 10.1109/ROMAN.2016.7745154. (Cited on page 81.)

[165] Saleha Raza, Sajjad Haider, and Mary-Anne Williams. Robot reasoning using first order bayesian networks. In Zengchang Qin and Van-Nam Huynh, editors, *Integrated Uncertainty in Knowledge Modelling and Decision Making*, pages 1–12, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39515-4. (Cited on page 38.)

[166] Joshua Reich, Vishal Misra, Dan Rubenstein, and Gil Zussman. Connectivity maintenance in mobile wireless networks via constrained mobility. *Selected Areas in Communications, IEEE Journal on*, 30(5):935–950, 2012. (Cited on page 104.)

[167] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, June 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2577031. URL `https://doi.org/10.1109/TPAMI.2016.2577031`. (Cited on page 3.)

[168] 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, April 2015. ISSN 1545-5955. doi: 10.1109/TASE.2014.2377791. (Cited on page 22.)

[169] F. Riccio, E. Borzi, G. Gemignani, and D. Nardi. Context-based coordination for a multi-robot soccer team. In Luis Almeida, Jianmin Ji, Gerald Steinbauer, and Sean Luke, editors, *RoboCup 2015: Robot World Cup XIX*, pages 276–289. Springer International Publishing, 2015. ISBN 978-3-319-29339-4. (Cited on pages 10, 58, 95, 97, and 154.)

[170] F. Riccio, E. Borzi, G. Gemignani, and D. Nardi. Multi-robot search for a moving target: Integrating world modeling, task assignment and context. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1879–1886, Oct 2016. doi: 10.1109/IROS.2016.7759298. (Cited on pages 11, 15, 38, 40, 41, 58, 95, 97, and 154.)

[171] F. Riccio, R. Capobianco, Marc Hanheide, and D. 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 pages 2, 9, 38, 58, and 154.)

[172] F. Riccio, R. Capobianco, and D. Nardi. Learning human-robot handovers through π-stam: Policy improvement with spatio-temporal affordance maps. In *International Conference on Humanoid Robots (HUMANOIDS16)*, 2016. (Cited on pages 11, 38, 58, 112, and 154.)

[173] F. Riccio, R. Capobianco, and D. Nardi. Using monte carlo search with data aggregation to improve robot soccer policies. In *Proceedings of the 20th International RoboCup Symposium*, 2016. (Cited on pages 11, 58, 112, and 154.)

[174] F. Riccio, A. Vanzo, V. Mirabella, T. Catarci, and D. Nardi. Enabling symbiotic autonomy in short-term interactions: A user study. In Arvin Agah, John-John Cabibihan, Ayanna M. Howard, Miguel A. Salichs, and Hongsheng He, editors, *Social Robotics - 8th International Conference, ICSR 2016, Kansas City, MO, USA, November 1-3, 2016, Proceedings*, volume 9979 of *Lecture Notes in Computer Science*, pages 796–807, Kansas City, MO, USA, November 2016. Springer International Publishing. ISBN 978-3-319-47437-3. doi: 10.1007/978-3-319-47437-3_78. (Cited on pages xii, 10, 58, 82, 88, and 154.)

[175] F. Riccio, R. Capobianco, and D. Nardi. Q-CP: Learning action values for cooperative planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages –, 2018. (Cited on pages 8, 11, 58, 112, and 154.)

[176] F. Riccio, R. Capobianco, and D. Nardi. LoOP: Iterative learning for optimistic planning on robots. *submitted to journal paper*, April 2018. (Cited on pages 11, 58, and 154.)

[177] F. Riccio, R. Capobianco, and D. Nardi. DOP: Deep optimistic planning with approximate value function evaluation. In *Proceedings of the 2018 International Conference on Autonomous Agents and Multiagent Systems*, pages –, 2018. (Cited on pages 8, 11, 38, 58, 112, and 154.)

[178] Francesco Riccio, Maria T. Lázaro, Guglielmo Gemignani, and Daniele Nardi. Multi robot perception and action: World modeling and task allocation. In *RSS Workshop on Principle of multi-robot systems*, 2015. (Cited on page 9.)

[179] Francesco Riccio, Roberto Capobianco, and Daniele Nardi. Using spatio-temporal affordances to represent robot action semantics. In *Machine Learning Methods for High-Level Cognitive Capabilities in Robotics Workshop, Workshop@IROS 2016*, 2016. (Cited on pages 8, 9, 58, and 154.)

[180] Laurel D. Riek. Wizard of oz studies in hri: A systematic review and new reporting guidelines. *J. Hum.-Robot Interact.*, 1(1):119–136, july 2012. ISSN 2163-0364. doi: 10.5898/JHRI.1.1.Riek. (Cited on page 86.)

[181] Raúl Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996. ISBN 3-540-60505-3. (Cited on page 121.)

[182] Stephanie Rosenthal and Manuela Veloso. Mobile robot planning to seek help with spatially-situated tasks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 2067–2073. AAAI Press, 2012. (Cited on page 82.)

[183] Stephanie Rosenthal, Joydeep Biswas, and Manuela M. Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, pages 915–922, 2010. doi: 10.1145/1838206.1838329. URL `http://doi.acm.org/10.1145/1838206.1838329`. (Cited on pages 7 and 82.)

[184] B. Rosman and S. Ramamoorthy. Action priors for learning domain invariances. *IEEE Transactions on Autonomous Mental Development*, 7(2):107–118, June 2015. ISSN 1943-0604. doi: 10.1109/TAMD.2015.2419715. (Cited on page 54.)

[185] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014. (Cited on pages 39, 56, and 118.)

[186] 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 113, 114, 116, 118, 121, 140, and 141.)

[187] S. I. Roumeliotis and G. A. Bekey. Distributed multirobot localization. *IEEE Transactions on Robotics and Automation*, 18(5):781–795, Oct 2002. ISSN 1042-296X. doi: 10.1109/TRA.2002.803461. (Cited on pages 41, 42, and 96.)

[188] 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 6.)

[189] Andrei A Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016. (Cited on pages 40 and 112.)

[190] H.G. Sage, M.F. De Mathelin, and E. Ostertag. Robust control of robot manipulators: A survey. *International Journal of Control*, 72(16):1498–1522, 1999. doi: 10.1080/002071799220137. URL `https://doi.org/10.1080/002071799220137`. (Cited on page 1.)

[191] A. Scalmato, A. Sgorbissa, and R. Zaccaria. Describing and classifying spatial and temporal contexts with owl dl in ubiquitous robotics. In *2012 IEEE International Conference on Robotics and Automation*, pages 237–244, May 2012. doi: 10.1109/ICRA.2012.6224835. (Cited on page 19.)

[192] 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 138.)

[193] J. Serafin and G. 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 20 and 36.)

[194] Y. Shapira and N. Agmon. Path planning for optimizing survivability of multi-robot formation in adversarial environments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4544–4549, Sept 2015. doi: 10.1109/IROS.2015.7354023. (Cited on page 41.)

[195] D. Shen, G. Chen, J. B. Cruz, and E. Blasch. A game theoretic data fusion aided path planning approach for cooperative uav isr. In *2008 IEEE Aerospace Conference*, pages 1–9, March 2008. doi: 10.1109/AERO.2008.4526563. (Cited on pages 15 and 17.)

[196] Thomas J. Shuell. Cognitive conceptions of learning. *Review of Educational Research*, 56(4):411–436, 1986. doi: 10.3102/00346543056004411. URL `https://doi.org/10.3102/00346543056004411`. (Cited on page 6.)

[197] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2164–2172. Curran Associates, Inc., 2010. URL `http://papers.nips.cc/paper/4031-monte-carlo-planning-in-large-pomdps.pdf`. (Cited on page 38.)

[198] David Silver, Richard S Sutton, and Martin Müller. Temporal-difference search in computer go. *Machine learning*, 87(2):183–219, 2012. (Cited on pages 112, 123, and 145.)

[199] 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 28, 38, and 39.)

[200] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. (Cited on pages 39, 43, and 113.)

[201] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1080–1087, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8. URL `http://dl.acm.org/citation.cfm?id=1643031.1643040`. (Cited on pages 5 and 38.)

[202] Reid G. Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan L. S. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 852–858. AAAI Press, 2000. ISBN 0-262-51112-6. URL `http://dl.acm.org/citation.cfm?id=647288.723404`. (Cited on page 38.)

[203] Hyang-gi Song, Michael Restivo, Arnout van de Rijt, Lori L. Scarlatos, David Tonjes, and Alex Orlov. The hidden gender effect in online collaboration: An experimental study of team performance under anonymity. *Computers in Human Behavior*, 50: 274–282, 2015. (Cited on page 92.)

[204] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, Jun 2000. ISSN 1573-7527. doi: 10.1023/A:1008942012299. URL `https://doi.org/10.1023/A:1008942012299`. (Cited on page 40.)

[205] Peter Stone and Manuela Veloso. Robot teams: From diversity to polymorphism. *A Survey of Multiagent and Multirobot System*, 2002. (Cited on page 96.)

[206] F. Stulp and S. Schaal. Hierarchical reinforcement learning with movement primitives. In *2011 IEEE-RAS International Conference on Humanoid Robots*, pages 231–238, Oct 2011. doi: 10.1109/Humanoids.2011.6100841. (Cited on page 138.)

[207] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981. (Cited on page 26.)

[208] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. (Cited on pages 115, 137, 138, and 140.)

[209] D. S. Syrdal, K. Lee Koay, M. L. Walters, and K. Dautenhahn. A personalized robot companion? - the role of individual differences on spatial preferences in hri scenarios. In *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, pages 1143–1148, Aug 2007. (Cited on page 92.)

[210] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. URL `http://arxiv.org/abs/1602.07261`. (Cited on page 120.)

[211] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. (Cited on page 29.)

[212] L. Takayama and C. Pantofaru. Influences on proxemic behaviors in human-robot interaction. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5495–5502, Oct 2009. (Cited on pages 88 and 92.)

[213] Moritz Tenorth and Michael Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System. *International Journal of Robotics Research (IJRR)*, 32(5):566 – 590, April 2013. (Cited on page 22.)

[214] 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 page 5.)

[215] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21 – 71, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(97)00078-7. URL `http://www.sciencedirect.com/science/article/pii/S0004370297000787`. (Cited on page 36.)

[216] Sebastian Thrun. Robotic mapping: A survey. In Gerhard Lakemeyer and Bernhard Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, chapter Robotic Mapping: A Survey, pages 1–35. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1-55860-811-7. (Cited on page 36.)

[217] Elena Torta, Raymond H. Cuijpers, and James F. Juola. A model of the user's proximity for bayesian inference. In *Proceedings of the 6th International Conference on Human-robot Interaction*, HRI '11, pages 273–274, New York, NY, USA, 2011. ACM. (Cited on page 86.)

[218] B. Tribelhorn and Z. Dodds. Evaluating the roomba: A low-cost, ubiquitous platform for robotics research and education. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1393–1399, April 2007. doi: 10.1109/ROBOT.2007.363179. (Cited on pages 1 and 15.)

[219] Roy M. Turner. Context-mediated behavior for intelligent agents. *Int. J. Hum.-Comput. Stud.*, 48(3):307–330, March 1998. ISSN 1071-5819. doi: 10.1006/ijhc.1997.0173. URL `http://dx.doi.org/10.1006/ijhc.1997.0173`. (Cited on pages 16 and 97.)

[220] 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 page 113.)

[221] Andrea Vanzo, Danilo Croce, Roberto Basili, and Daniele Nardi. Structured learning for context-aware spoken language understanding of robotic commands. In Mohit Bansal, Cynthia Matuszek, Jacob Andreas, Yoav Artzi, and Yonatan Bisk, editors, *Proceedings of the First Workshop on Language Grounding for Robotics, Vancouver, Canada, August 3, 2017.*, pages 25–34, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2804. URL `http://www.aclweb.org/anthology/W17-2804`. (Cited on page 1.)

[222] R. Vazquez-Martin, P. Nunez, A. Bandera, and F. Sandoval. Spectral clustering for feature-based metric maps partitioning in a hybrid mapping framework. In *2009 IEEE International Conference on Robotics and Automation*, pages 4175–4181, May 2009. doi: 10.1109/ROBOT.2009.5152476. (Cited on page 36.)

[223] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017. (Cited on page 112.)

[224] Arun Venkatraman, Roberto Capobianco, Lerrel Pinto, Martial Hebert, Daniele Nardi, and James A. Bagnell. Improved learning of dynamics models for control. In *2016 International Symposium on Experimental Robotics*, 2016. (Cited on page 39.)

[225] Michael L Walters, Kerstin Dautenhahn, René Te Boekhorst, Kheng Lee Koay, Dag Sverre Syrdal, and Chrystopher L Nehaniv. An empirical framework for human-robot proxemics. In *Proceedings of the Symposium on New Frontiers in Human-Robot Interaction*, pages 144–149, Edinburgh, Scottland, 2009. (Cited on page 92.)

[226] 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 pages 38 and 48.)

[227] Yizao Wang and Sylvain Gelly. Modifications of UCT and sequence-like simulations for monte-carlo go. In *2007 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2007. doi: 10.1109/cig.2007.368095. URL `https://doi.org/10.1109%2Fcig.2007.368095`. (Cited on page 114.)

[228] Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017. (Cited on page 136.)

[229] David Wingate, Noah D Goodman, Daniel M Roy, Leslie P Kaelbling, and Joshua B Tenenbaum. Bayesian policy search with policy priors. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1565, 2011. (Cited on page 54.)

[230] T. Witzig, J. M. Zöllner, D. Pangercic, S. Osentoski, R. Jäkel, and R. Dillmann. Context aware shared autonomy for robotic manipulation tasks. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5686–5693, Nov 2013. doi: 10.1109/IROS.2013.6697180. (Cited on page 19.)

[231] Jay Young, Valerio Basile, Markus Suchi, Lars Kunze, Nick Hawes, Markus Vincze, and Barbara Caputo. Making sense of indoor spaces using semantic web mining and situated robot perception. In Eva Blomqvist, Katja Hose, Heiko Paulheim, Agnieszka Ławrynowicz, Fabio Ciravegna, and Olaf Hartig, editors, *The Semantic Web: ESWC 2017 Satellite Events*, pages 299–313. Springer International Publishing, 2017. ISBN 978-3-319-70407-4. (Cited on page 1.)

[232] 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 3, 6, and 37.)

[233] Bin Zhao, Yongqiang Guan, and Long Wang. Controllability improvement for multi-agent systems: leader selection and weight adjustment. *International Journal of Control*, 89(10):2008–2018, 2016. doi: 10.1080/00207179.2016.1146970. URL `https://doi.org/10.1080/00207179.2016.1146970`. (Cited on page 40.)

[234] Kaiyu Zheng, Andrzej Pronobis, and Rajesh P. N. Rao. Learning Graph-Structured Sum-Product Networks for probabilistic semantic maps. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, LA, USA, February 2018. URL `http://www.pronobis.pro/publications/zheng2018aaai`. (Cited on page 36.)

[235] X. S. Zhou and S. I. Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1785–1792, Oct 2006. doi: 10.1109/IROS.2006.282219. (Cited on pages 41 and 96.)

[236] Yan Zuo and Tom Drummond. Fast residual forests: Rapid ensemble learning for semantic segmentation. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 27–36. PMLR, 13–15 Nov 2017. URL `http://proceedings.mlr.press/v78/zuo17a.html`. (Cited on page 38.)