Universitat Politècnica de Catalunya

Ph.D. Program:

AUTOMATIC CONTROL, ROBOTICS AND COMPUTER VISION

Ph.D. Thesis

Knowledge Representation and Reasoning for Perception-based Manipulation Planning

Mohammed Diab

Thesis Advisor: Jan Rosell Gratacòs

November 2020

Knowledge Representation and Reasoning for Perception-based Manipulation Planning

Submitted in partial fulfillment of the requirements for the degree of Ph.D. in

Automatic Control, Robotics and Computer Vision

Supervised by

Jan Rosell Gratacòs

Institut d'Organització i Control de Sistemes Industrials

Universitat Politècnica de Catalunya

November 2020

This thesis is dedicated to the spirit of my father, my children, and my sisters' children.

Acknowledgments

I would like to thank the Responsible of the Doctoral programme in Automatic Control, Robotics and Computer Vision Prof. Raúl Suárez for his efforts and guidance during the Ph.D. My external collaborators Prof. Micheal Beetz, Prof. John Bateman, Dr. Mihai Pomarlan and Daniel Beßler from Bremen University, Prof. Florentin Wörgötter, Dr. Tomas Kulvicius from Georg-August-Universität Göttingen, and finally Prof. Stefano Borgo from Laboratory for Applied Ontology (LOA), ISTC CNR, Trento, Italy, for their guidance during my research stays.

A very special thanks should be presented to my supervisor Prof. Jan Rosell for his guidance, motivation, patience, discussion, and help in all aspects to complete this thesis. I would also like to thank Dr. Ali Akbari and Dr. Muhayy Ud Din for their help and useful long discussion. I would also like to thank all the members of the Service of Industrial Robotics research group for their help. I would like to thank Leopold Palomo for his guidance in the implementation stuff.

A very special thanks to my mother, Wahida El-Husseiny, my wife, Fatema Mohamed, my sisters, and the rest of my family for there infinite support, it was not possible to complete the thesis without their support in all aspects.

All praise and glory be to God who is the greatest benefactor, and Whose helps enabled me to complete this thesis.

Contents

List of Figures

List of Tables

Abstract

This thesis addresses the perception-based knowledge representation and reasoning for a combination of task and motion planning to deal with different types of robotic manipulation problems, ranging from single or multiple collaborative mobile robots navigating among movable obstacles to complex higher-dimensional table-top manipulation problems carried out by dual-arm robots or mobile manipulators. For those problems, besides the combination of task and motion levels of planning, the integration of perception models with knowledge to guide both planning levels, resulting in a sequence of actions or skills which, according to the current knowledge of the world, may be executed, is necessary. This combination pursues the obtention of a geometrically feasible manipulation plan through a symbolic and geometric search space. It has emerged as a challenging issue as the failures due to geometric constraints lead robots to dead-end tasks.

Manipulation tasks in which there may be interactions between robots and objects are considered. To cope with them, knowledge about the physics of the environment is integrated with a combination of task planning and physics-based motion planning, allowing to deal with push and pull actions and also to look for low-cost plans in terms of power. This integration enriches the planning process and to aid in providing ways of executing symbolic actions.

Problems with uncertain information (in the initial state of the robot world or in the result of symbolic actions) and problems where humans and robots interact are also considered. To deal with such issues, a combination of contingent-based task and motion planner with knowledge for failure interpretation and recovery is proposed, which assumes the availability of a perception system (to evaluate the actual state of the environment) and the collaboration of the human operator (robots can ask humans for those tasks which are difficult or infeasible for them).

For every-day tasks, experiential knowledge can play a significant role to make the robot capable to learn from its experience instead of repeatedly planning the same task with the same givens within the planning system, which could be computational and time-consuming process. To deal with such issues, a robotics framework is proposed which is equipped with a module with experiential knowledge (learned from its experience or given by the user) on how to execute a set of actions, like pick-up, put-down, or open a drawer, using workflows as well as robot trajectories.

An implementation framework to combine different types of task and motion planners is presented. All the required modules and tools are illustrated, including the explanations on the flow of information between the different languages used, Prolog and C++.

Chapter 1

Introduction

This chapter describes the structure and contents of this thesis. It contains the problems that this thesis deals with, as depicted in Sec. 1.1, proposed solutions, as depicted in Sec. 1.2, and the thesis contributions, Sec. 1.3. The proposed solutions require approaches to be used such as planning, knowledge representation, reasoning, perception, as well as learning, as described in Sec. 1.4. These approaches have been used and integrated together in order to increase robot autonomy. Several challenges have been set to test the proposed solutions, as depicted in 1.5. These solutions have been published in robotics and Artificial intelligence (AI) journals, as well as conferences, they are mentioned in Sec. 1.6. These works have been done with collaboration with other research centers, as mentioned in Sec. 1.7.

1.1 Problem statement

Indoor robots with autonomy, mobility and manipulation capabilities have the potential to act as robot helpers at home to improve the quality of life for various user populations, such as elder and handicapped people, or to act as robot co-workers at factory floors, helping for instance in assembly applications where collaborating with other operators may be required. In these semi/unstructured environments, the robot task may not be properly organized and the model of the environment completely known, and therefore abundant problems have to be taken into consideration, for example, the need of removing obstacles in order to have precisely access to a particular object, which requires the integration of symbolic and geometric planning levels, with skills such as pick-up, put-down or navigate. Moreover, perception systems using vision or depth sensors may be required to model the geometry and the pose of objects in the environment and regularly update their status. Different types of sensors have their own limitations, so perception systems based on multi-sensory data integration combining information from different sources are very useful to obtain information, which in some sense is better than the one obtained when

the sources are used separately.

To plan the tasks to be done by the robot in the above-mentioned scenarios, sophisticated planning mechanisms are required to adapt the actual state of the environment and comply with constraints both at task and at geometric levels. There are two dominant approaches in the manipulation planning domain, one based on classical task planning and the other based on knowledge and reasoning. The former mainly uses the Planning Domain Definition Language (PDDL) to describe the world. The main advantage of this way of description is that it can easily handle tasks with many actions, and integrate the geometric (motion) constraints. However, it makes the closed world assumption, i.e., if some facts about the world are not known or change, a planner may not be able to find a solution. This limitation means that robots are not able to begin a task until all objects in the environment are known and the actions the robot can do on them are completely defined. The latter has emerged as a new domain of planning, focused on making the robot able to flexibly perform manipulation tasks. The main advantage of this approach is that it can easily integrate the knowledge from the environment and adapt the action to be done accordingly. However, in complex manipulation problems that may have many task (geometric) constraints, such as the Towers of Hanoi problem, where task and motion levels are coupled, they may fail to compute a long sequence of actions with feasible motion solutions (Lagriffoul et al., 2018), (Beßler et al., 2018a).

1.2 Proposed solutions

To tackle the aforementioned limitations of both approaches, some components are required that may facilitate the process of manipulation planning. First, a perception system to perceive the objects and their features, and a mechanism to capture the semantics of the actual scene and prepare the planning accordingly. Second, a mechanism to incorporate geometric knowledge and motion planning. Finally, a knowledge-based inference mechanism to reason on relative positioning, preconditions satisfaction and action feasibility. That is, the integration of perception and knowledge with manipulation planning approaches, may help to cover some missing components such as reasoning mechanisms able to analyze the feasibility of actions, the availability of placement regions, the reachability of grasping motions, and the satisfaction of manipulation constraints. These components play a significant role in robotic manipulation, specially for bi-manual robot tasks or for multi-robot cooperation, that require geometric reasoning, including physics-based motion planning with queries about how interaction with the objects is to be done. Also, knowledge-based sensing modules can be integrated with planners coping with uncertain scenarios, that require reasoning about the sensing system, the features of the environment entities, and sensor limitations. This way, the robot can be aware about the type of sensors that it has and how to use them.

Hence, knowledge-based reasoning is proposed to facilitate the process of manipulation. This solution has the capability to make the world open to cope with environmental dynamic entities. It can be used for complex manipulation tasks that require the combination of both

symbolic and geometric levels of planning and the integration with a perception module can provide a rich semantic description for the robot whenever needed. In this sense, a well-structured knowledge representation plays a significant role. Many ways are used for knowledge representation, such as ontologies, that are concerned with structuring concepts and relations such that they are usable for reasoning tasks done by artificial systems (e.g., robots). Formally, an ontology is defined as “an explicit, formal specification of a shared conceptualization" (Gruber, 1995). The conceptualization refers to the abstract models of entities in a certain domain. These models are achieved by defining their relevant concepts along with their relations.

In this line, this thesis develops a series of modeling and reasoning tools for knowledge-oriented manipulation planning in semi/unstructured environments. The main idea is to use high-level knowledge-based reasoning to capture a rich semantic description of the scene, knowledge about the physical behavior of the objects, and inference mechanism to reason about the potential manipulation actions. Moreover, a multi-sensory module is proposed to perceive the objects in the environment and build the ontological knowledge.

1.3 Contributions

Mobile manipulators acting as robot co-workers are required to work autonomously in human environments, and in the presence of human operators. Autonomy can be achieved with integrated task and motion planning capabilities, that are able to find feasible plans for the robot to execute complex tasks. Perception capabilities are, however, a key issue for the successful execution of tasks, because human environments are semi-structured and affected by uncertainty. Therefore, a perception module with different sensors, including those like RFID that can cope with non-line-of-sight (NLOS) situations, as well as sensory fusion mechanisms, is required. The availability of such a perception module may allow to consider sensing actions in the task planning procedure, to reduce the effects of uncertianty in the initial state and in the actions effects. Moreover, to be able to face difficult manipulation tasks in these semi-strucuted environments, some semantic knowledge on the objects of the environment and on the possible manipulation actions, is required. This knowledge may guide both the planning at motion level and at task level.

This thesis contributes in this line by implementing and integrating the necessary module to increase the robot autonomy. The contribution are listed below.

- *Knowledge Guidance for Task and Motion Planning:* A manipulation knowledge framework, called Knowledge-based Task and Motion Planning (KTAMP) is presented. The framework contains the tool called, Perception and Manipulation Knowledge (PMK) which is presented in terms of an ontology-based modeling and reasoning process. PMK aims at being shared and reused, and for this, PMK ontology relies on other upper and reference/domain ontologies: the Suggested Upper Merged Ontology (SUMO) (Niles and

Pease, 2001a) and the Core Ontology for Robotics and Automation (CORA) (Prestes et al., 2013). The reasoning mechanism includes some reasoning processes for autonomous robots to enhance Task and Motion Planning (TAMP) capabilities in the manipulation domain. A perception module can be integrated with the framework to capture a rich semantic description of the scene, knowledge about the physical behavior of the objects, and reasoning about the potential manipulation actions. The reasoning scope of PMK is divided into four parts: reasoning for perception (e.g. which perceptual features can be obtained and with which sensors?), the reasoning for object manipulation features (e.g. how can a given object be manipulated according to its characteristics and the current pose), the reasoning for a situation, (e.g. which are the spatial relations of the objects in the scene?), and reasoning for planning (e.g. can a given primitive be applied at the current scene?). The PMK tool is also integrated with physics-based motion planning that aims mainly to provide the way of interactions between a robot and objects holding specific manipulation constraints. Specially, PMK provides:

1. *Standard representation:* The knowledge modeling is proposed by adapting the available concepts provided by IEEE-1872 standards of knowledge representation for the robotic domain. Moreover, some uncovered concepts related to manipulation domains have been proposed, such as knowledge related to sensors.
2. *Knowledge representation for perception:* The perception ontology is proposed to include the perceived information from different sensors, e.g., the representation is workable for cameras or Radio Frequency Identification (RFID), and may include any implemented sensing library.
3. *Situation analysis:* Inference process predicates are developed based on Description Logic (DL) to evaluate the objects' situation in the environment based on spatial reasoning, and to relate the classes entities and reason over them. Moreover, potential placement region and spatial reachability of the robot are introduced.
4. *Planning enhancement:* The use of PMK as a black-box allows any planner to reason about TAMP requirements, such as robot capabilities, action constraints, action feasibility, and manipulation behaviors. That includes a semantic extension to automatically construct and categorize the objects into different types according to the objects and task constraints. Also, the interaction dynamics extension to define a knowledge that allows the planner to deal with interaction dynamics.

- *Heterogeneous reasoning planning approach:* An ontology-based framework for failure interpretation and recovery in planning and execution called FailRecOnt is proposed towards a more automated, reasoning and knowledge-driven approach to failure handling. Such an approach would need a concept of what "failure" means, what kinds of failures might happen and why, and concepts to define what an appropriate response might be. The reasoning must also be integrated into the geometric ontology that gives access to geometric reasoning such as collision check to check the actions' feasibility as well as the perception action loop of the robot, and able to guide a plan repair process to resume or repeat a task after a failure. In more detail, FailRecOnt provides:

  1. *Ontology formulation:* Formal definition of concepts to describe failures according to aspects such as causal mechanism, location, time relative to task performance, and

functional considerations e.g. resources, and concepts to describe recovery strategies according to the plan repair operations they require.

2. *Modeling in different foundations:*
   - *For robotics domain:* Modeling the absolute abstract concepts under robotics upper-level ontologies such as CORA, which uses the SUMO ontology as an upper level.
   - *For engineering domain:* Modeling the absolute abstract concepts under very generic foundational ontologies such as DUL.
3. *Integration of geometric reasoning module within ontology:* Integration of how to call the low-level geometric reasoning such as collision check, motion planning, inverse kinematic (IK) and object placement from ontology, which is a step toward more robot autonomy.
4. *Use of the failure ontology:* Description of how to use the failure ontology in a task and motion planning (TAMP) process using a knowledge-driven approach, resulting in an heterogeneous reasoning planning approach.

- *Skill-based task and motion planning:* A Skill-based Robotic Manipulation Framework based on Perception and Reasoning framework, called SkillMaN is proposed, which is equipped with a module with experiential knowledge (learned from its experience or given by the user) on how to execute a set of skills, like pick-up, put-down or open a drawer, using workflows as well as robot trajectories. The framework also contains an execution assistant with geometric tools and reasoning capabilities to manage how to actually execute the sequence of motions to perform a manipulation task (which are forwarded to the executor module), as well as the capacity to store the relevant information to the experiential knowledge for further usage, and the capacity to interpret the actual perceived situation (in case the preconditions of an action do not hold) and to feedback the updated state to the planner to resume from there, allowing the robot to adapt to non-expected situations. Aiming at giving the robot more autonomy, specifically, SkillMaN framework provides the following services and modules:

  1. *Perception:* An integrated multi-sensory module based on RFID sensors, with storage data capability, and RGB-D cameras,
  2. *Situation similarity check:* Service to semantically check the situation similarity based on robot goals and perception outcomes. It helps the robot to decide whether to use its experience-based knowledge,
  3. *Planning:* Service to interleave symbolic and geometric reasoning levels with the perception module to make the robot capable to partially figure out the environment and plan under partial information, and
  4. *Experiential knowledge:* Module to store experiential knowledge on how to execute a set of skills, like pick-up, put-down or open a drawer, to be used for instance for adapting the motion of the robot in similar situations.

  All these services and modules are used through a task manager in a way that can be smoothly used in different tasks.

- *Implementation framework.* To implement the aforementioned frameworks, several implementation tools are proposed. These tools are implemented using Protégé editor for modeling ontologies, C++ and Python for grounding information and the Prolog languages for inference mechanism, all need different integration techniques. Hereupon, the implementation provides a flexible way to handle the communication between the languages using mainly *Robotic Operating System* shell and the *SWI-Prolog* library. It enables to execute the frameworks in a simulation and in a real environment. Besides the modeling of the proposed ontologies and inference mechanisms, a multi-sensory system based on Vision and Radio Frequency Identification (RFID) technology is implemented which provides some ROS-based services to enrich the perception of the environment. The perception part has been used in the implementation of the contingency plan presented in (Akbari et al., 2020). Moreover, a reasoning action is used to monitor the manipulation actions in case of failure occur. Specifically, these actions are used for:

    1. *State interpretation based on observation:* Integration of perception and the knowledge modeling of failures.
    2. *Plan repair guided by reasoning:* Failure descriptions are the input of reasoning processes that map failure types to possible recovery strategies via classification reasoning. The conceptualization of recovery strategies present in the failure ontology allows posing repair and replanning queries to a contingency task planner.
    3. *Communication and requesting assistance:* One of the recovery strategies we have included in the ontology is delegating a difficult task to another agent, and in particular asking a human observer for help. A reasoning based approach that accounts for the robot's capabilities allows deciding when such a strategy is necessary and what should be communicated to the human from whom assistance is requested.

1.4 Thesis roadmap

Fig. 1.1 illustrates the contents of the three main chapters. The first part of the thesis, described in chapter three, is focused on the techniques to provide useful knowledge to guide and facilitate the planning process within a classical-based manipulation planning framework. This planning framework facilitates the combination of task and motion planning (TAMP) approaches which includes Fast Forward (FF), a classical symbolic planning approach to compute the sequence of actions to be done in a certain task, and physics-based motion planning which deals with motions and possible interactions with the objects. The tool proposed to provide useful knowledge to the planning process is called Perception and Manipulation Knowledge (PMK). It provides, on the one hand, a standardized formalization under several foundations, such as SUMO, and CORA, in order to facilitate the shareability and reusability when the interaction between humans and/or robots is done. On the other hand, it provides the inference mechanism to reason about TAMP requirements, such as robot capabilities, action constraints, action feasibility, and manipulation behaviors. Moreover, PMK allows to break the closed world

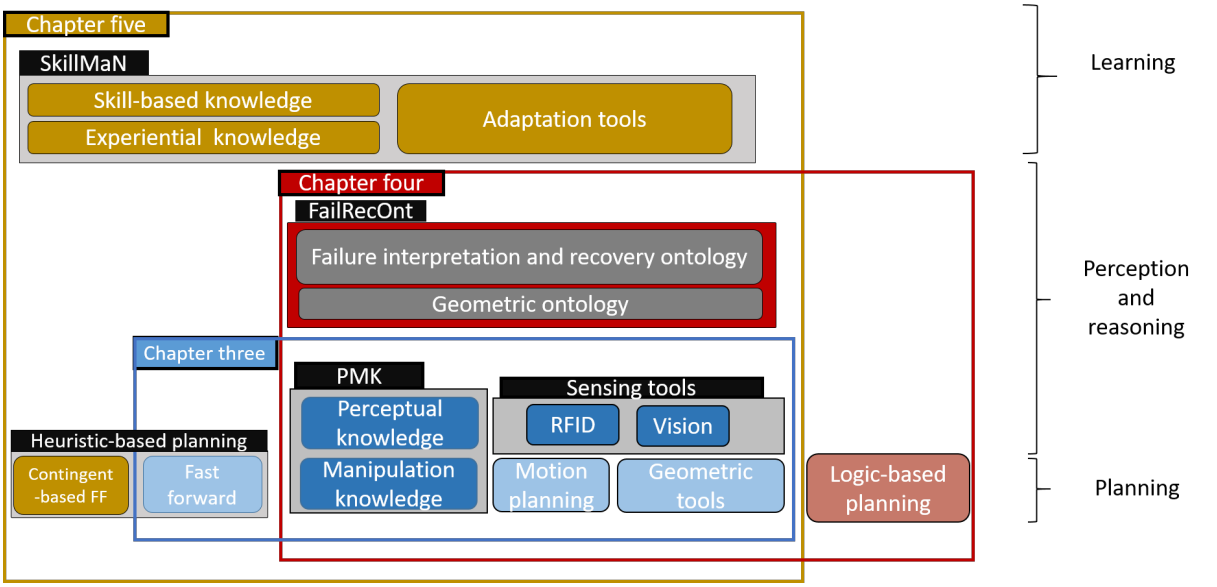


Figure 1.1: Schematic view of the thesis with the proposal to make the robots incrementally more autonomous. In blue the contribution of chapter three with the perception and manipulation knowledge (PMK) to guide the planning. In red the contribution of chapter four with the failure and recovery ontology (FailRecOnt) to achieve robust behavior. In brown, the contribution of chapter five with the SkillMaN framework to adapt to new situations by the re-use of previous knowledge. Light-colored modules are these where there is no direct contribution of this thesis, either because they are open-source available packages or they have been developed by other research groups within research collaboration.

assumption of classical-based manipulation planning approaches. This proposal has been tested for a serving task in a table-top manipulation problem.

The second part of the thesis, described in chapter four, is focused on providing FailRecOnt framework which provides a failure interpretation and recovery knowledge for different planner approaches. We are mainly interested in assembly manipulation tasks for bi-manual robots, which often encounter complexity or failures in the planning and execution phases. Planning phase failures typically refer to failures of the planner itself, but we will use planning phase failures to also refer to situations where the planner reasons that some action would be infeasible, e.g. because objects block access to what the robot should reach. A correct selection of grasps and placements must be produced in such an eventuality. Depending on the type of problem, goal order must be carefully handled especially in the assembly domain; very large search spaces are possible, requiring objects to be moved more than once for achieving the goals. Execution phase failures refer to hardware failures related to the system devices, e.g. robot or camera needs to be re-calibrated, or software failures related to the capabilities offered by specific software components, or failures in action performance such as an unexpected occluding object, or slippage.

Also, the description of how to use the failure ontology in a task and motion planning (TAMP) process, such as the knowledge-driven approach is provided. A logic-based planning approach is used, provided by KnowRob group at University of Bremen (Beßler et al., 2018b), which represents assembly recipes in description logic theories. With such a recipe, the robot can figure out the next assembly step through logical inference. However, before performing an action, the robot needs to ensure that various spatial constraints are met, such as that the parts to be put together are reachable, non occluded, etc. Such inferences are very complicated to support in logic theories, but specialized algorithms exist that efficiently compute qualitative spatial relations such as whether an object is reachable. Here, a heterogeneous reasoning is proposed. The heterogeneous reasoning process requires the integration between the knowledge-based assembly planning level and the geometric level, which includes modules to infer about the spatial relations and feasibility of actions. These geometric modules feed back the ontological planner about the result of the reasoning processes, such as testing the reachability or the existence of a collision-free path, done using geometric, failure interpretation and recovery ontologies. These ontologies are used together to provide a strategy of sequentially calling the low-level geometric modules (i.e., motion planning, collision check, object placement and inverse kinematics). Also, they are used for the interpretation process to figure out the type of failures and their source(s), if they occur, and based on this interpretation prepare a potential recovery process.

The third part of the thesis, described in chapter five, is focused on the use of knowledge-based experience, called experiential knowledge, in every-day manipulation problems, such as, in service robotics applications, *serve* a cup in a cluttered environment, where some repeatable skills are usually used like pick-up, place-down or navigate. To efficiently handle these tasks, instead of entirely planning them every time (which could be computationally expensive), the integration of adaptation modules to adapt those skills in different situations is proposed. In this sense, a planning and execution framework for robotic manipulation tasks is presented, which is equipped with a module with experiential knowledge (learned from its experience or given by the user) on how to execute a set of skills, like pick-up, put-down, navigate or open a drawer, using workflows as well as robot trajectories. The framework also contains: an execution assistant with geometric tools, reasoning capabilities to manage how to actually execute the sequence of motions to perform a manipulation task (which are forwarded to the executor module), the capacity to store the relevant information to the experiential knowledge for further usage, and the capacity to interpret the actual perceived situation (in case the preconditions of an action do not hold) and to feed back the updated state to the planner to resume from there, allowing the robot to adapt to non-expected situations.

1.5 Motivation examples

Different challenging examples in the manipulation domain have been set to show the usefulness of all the proposal made.

- **Table-top manipulation problems**: The need to reason on the current state of the world in order to apply one action or another (like pick or push an object to take it apart) in a table-top manipulation scenario. KTAMP framework offers the way to encode the knowledge and reason upon it.

- **Assembly manipulation problems**: The need to reason on the result of actions or the need to evaluate actions preconditions (like the feasibility of a given grasp to execute a pick action) in an assembly task. FailRecOnt framework offers a flexible reasoning tool that is perfectly adapted to knowledge-driven planning schemes.

- **Mobile-based every-day manipulation problems**: The need to make the robot aware of situation similarities to efficiently re-use previous known ways to execute actions by adapting the robot motions to stored patterns (like opening one drawer once the robot knows how to open another). The SkillMaN framework integrates the previous tools with a similarity situation evaluator and a motion adaptation tool.

To validate the KTAMP framework, a motivation example is included that introduces a manipulation table-top problem at IOC-Lab. Two robots, the dual-arm YuMi robot and the TIAGo mobile manipulator, share the knowledge between each other in order to perform the task of serving beverages on a table. The set-up is prepared by YuMi, while TIAGo is used to actually serve the beverages to the customers.

To validate the FailRecOnt framework, an example describes a task of assemble a Battat toy under laboratory conditions where all parts of the toy are homogeneously distributed on a table. The YuMi robot is used to put together the body of the plane toy which has two wings and two parts that form a cylindrical body.

An example describes a task to classify an object in a given tray according to its color. The TIAGo robot with navigation capability is used where the object may be among other objects in cluttered scene, which have to be manipulated according to its nature.

To validate the SkillMaN framework, the mobile manipulator TIAGo is assumed to work in the semi-structured environment, where there are several *cans* (some filled and some empty) and a file cabinet with three drawers (the first to store empty cans and the second to store the filled ones). Two tasks are scheduled, one task is to sort the cans and store them on the corresponding drawers, and another one is to take a stored filled can and serve its contents to a customer. A perception system is used to figure out the location and status of the objects. To execute these tasks, some skills are introduced: *pickUp*, *putDown*, *openDrawer* and *serving*. Moreover, to integrate the FailRecOnt with SkillMaN, an example is proposed describing a task to classify an object in a given tray according to its color. The TIAGo robot with navigation capability is used where the object of interest is placed in cluttered scene among other objects that need to be manipulated differently according to their nature.

1.6 List of Publications

The list of all publications which I participated as author or co-author is the following:

Journal Publications:

1. Diab, M., Akbari, A., Ud Din, M., Rosell, J.: PMK - a knowledge processing framework for autonomous robotics perception and manipulation. Sensors 19 (5) (2019).
2. Olivares-Alarcos, A., Beßler, D., Khamis, A., Goncalves, P., Habib, M.K., Bermejo-Alonso, J., Barreto, M., Diab, M., Rosell, J., Quintas, J., et al.: A review and comparison of ontology-based approaches to robot autonomy. The Knowledge Engineering Review 34 (2019).
3. Akbari, A., Diab, M., Rosell, J.: Contingent task and motion planning under uncertainty for human-robot interactions. Journal of Applied Sciences (2020), section: Mechanical Engineering, special issue: Multi-Robot Systems: Challenges, Trends and Applications.
4. Diab, M., Pomarlan, M., Beßler, D., Akbari, A., Rosell, J., Bateman, J., Beetz, M.: Skill-based robotic manipulation based on perception and reasoning. Journal of Robotics and Autonomous Systems, Volume 134, 2020.
5. Beßler, D., Pomarlan, M., Bateman, J., Beetz, M., Diab, M., Akbari, A., Ud Din, M., Rosell, J.: Ontology guided planning and execution for flexible and robust low volume manufacturing. Conditionally accepted: Journal of Artificial Intelligence Research (2020).

Conference Publications:

1. Diab, M., Akbari, A., Muhayyuddin, and Rosell, J.: An ontology framework for physics-based manipulation planning. Published in Iberian Robotics conference, Springer (2017) 452-464.
2. Beßler, D., Pomarlan, M., Akbari, A., Muhayyuddin, Diab, M., Rosell, J., Bateman, J., Beetz, M.: Assembly planning in cluttered environments through heterogeneous reasoning. In Trollmann, F., Turhan, A.Y., eds.: Published in: KI 2018: Advances in Artificial Intelligence, Cham, Springer International Publishing (2018) 201-214.
3. Diab, M., Pomarlan, M., Beßler, D., Akbari, A., Rosell, J., Bateman, J., Beetz, M.: An ontology for failure interpretation in automated planning and execution. Published in Iberian Robotics conference, Springer (2019) 381-390.
4. Diab, M., Pomarlan, M., Stefano, B., Beßler, D., Rosell, J., Bateman, J., Beetz, M.: FailRecOnt - An Ontology-based Approach for Failure Interpretation and Recovery in Planning and Execution. Submitted to AAMAS 2021: International Joint Conference on Autonomous Agents and Multi-agent Systems.

1.7 Publication Note

Chapter three is partially covered in journal papers 1 and 2 and conference paper 1. Chapter four is partially covered in journal paper 5 and conference papers 2 and 3. Chapter five is partially covered in journal papers 3 and 4 and conference paper 4.

Joint Works: Two joint works have been done with the collaboration with other research centers. The first one has been done in the collaboration with the *KnowRob* group from institute for artificial intelligence (IAI), Bremen university. The geometric ontology and failure interpretation module proposed for combining task and motion planning for bi-manual robots has been offered to a knowledge-based planner developed by *KnowRob* group. The result is presented in journal paper 5 and conference papers 2 and 3. Moreover, in conference paper 4, a reasoning action has been integrated to the contingent-based task and motion planning presented in journal paper 3 to monitor the result of manipulation actions. In this work, a collaboration with the head of Laboratory for Applied Ontology (LOA), ISTC CNR, Trento, Italy has been done for modeling enhancement of the ontology. The second is the ongoing collaboration with the George-August-Universität Göttingen to develop a Learning-based technique to enhance the robot manipulation in indoor environment. The work is going to be submitted in a journal of robotics and autonomous system. Eventually, as a parallel work, the collaboration with the IEEE ORA-WG (Ontology for Autonomous Robotics -Working Group) on the new standard IEEE 1872.2 on IEEE standard ontologies for robotics and automation is in progress. This proposal enhances the old version of the standard IEEE 1872-2015 by modeling new concepts such as Function, Behavior, Capability and others. A case study is presented here, that includes the collaboration tasks between two robots, to validate the new standardized abstract concepts.

Chapter 2

Related Work

This chapter focuses on state-of-the-art researches in manipulation planning, standardized knowledge representation, and perception-based semantic knowledge in the robotics domain, as well as the integration between them to increase robot autonomy. Moreover, a logic-based planning approach is described in more detail than other approaches because some of the ontologies proposed in this thesis are naturally integrated there.

2.1 Manipulation planning

Manipulation problems are referred to as problems in which robots manipulate objects using a set of primitives, e.g., pushing, picking, or placing. Due to task constraints, the limitation of generic motion planning emerges, and the robot is required to displace objects when there is no feasible solutions between two robot configurations.

A more general manipulation planning approach has been developed by (Siméon et al., 2004) that considers multiple possible grasps (that can be used for re-grasping the objects) and stable placements of the movable objects to solve the problem. In the related field of grasp planning, (Azizi et al., 2017) propose a geometric approach based on detecting a complete set of object subsurfaces in a cluttered scene, which allows the end-effector to safely approach and grasp the object. In a similar way, (Hertle and Nebel, 2017) present techniques for sampling appropriate geometric configurations for object placements, grasping poses, or robot positions, in order to perform a specific action. And recently, a method for grasp planning in cluttered environments (Muhayyudin et al., 2018) has been proposed, which uses randomized physics-based motion planning to account for robot-object and object-object interactions. This allows a robot to push obstructing objects away while reaching a target grasp pose.

A manipulation planning framework with perception capability has been proposed (Migimatsu and Bohg, 2020) that optimizes over Cartesian frames defined relative to target objects. The resulting plan remains valid even if the objects are moving and can be executed by reactive controllers that adapt to these changes in real time. The framework is applied to a torque-controlled robot in a pick and place setting and demonstrate its ability to adapt to changing environments, inaccurate perception, and imprecise control, both in simulation and the real world. As a complementary of this work, a learning technique has been proposed in (Shao et al., 2020) to endow a robot with the ability to learn manipulation concepts that link natural language instructions to motor skills. In (Englert and Toussaint, 2018) learning manipulation skills from a single demonstration is proposed where a robot is demonstrated a manipulation skill once and should then use only a few trials on its own to learn to reproduce, optimize, and generalize that same skill.

The manipulation problem of *Navigation in indoor cluttered environment*, at which a robot needs to manipulate some objects in the workspace in order to reach its goal region, has been addressed by in (Stilman et al., 2007; Stilman and Kuffner, 2008) and (Hauser and Latombe, 2010; Hauser et al., 2010; Hauser, 2014) considering the existence of the objects occluding the way between two robot configurations.

The reviewed works concentrate on geometry challenges in manipulation with no high-level reasoning, while there are other manipulation problems which require to search in a symbolic search space to find a sequence of actions. The approaches, furthermore, lack from underlying discrete symbolic relations (e.g. among robots and objects) which can break down the complexity of the problem, understood as the need to check the feasibility of the actions used in a certain task. Therefore, combining task and motion planning makes a significant role to deal with different challenges of manipulation problems for robots.

2.1.1 Task planning

There has been a significant amount of studies in task planning, comprising a variety of different automated task planning approaches such as hierarchical, plan graph, and heuristic planning. Hierarchical approach (Wolfe et al., 2010), (Kaelbling and Lozano-Pérez, 2011) establishes a network to assign possible preconditions of actions (primitive or compound). To find a plan, tasks are decomposed and grow in a search space until satisfying primitives actions. The plan graph (Bryce et al., 2006), (Blum and Furst, 1997) constructs a search space of plans with the form of a graph and establishes a sequence of levels gradually expanded through the search space. The main merits of graph-based planners are the ability to analyze combinations of action to satisfy the task, and the ability to evaluate multiple ways to reach the goal. Regarding heuristic planning, one of the successful approach is FF (fast forward). It has two main components: RPG (relaxed planning graph), and state space search. RPG is the simplified version of the plan graph, i.e., delete list (the facts which are deleted by action) of actions are ignored. The latter is devoted to the search for the highest chance of success using the heuristic values, i.e. helpful actions (those actions that executed from a state have a high chance of being

in the final plan) that it uses to guide the search of the state space (Dearden and Burbridge, 2014). To handle uncertainty in task planning (such as the uncertainty in the action effect), variants of FF such as contingent, and conformant FF have been proposed. Contingent FF is the task of generating a conditional plan given uncertainty about initial condition and action effects, but with the ability to observe some aspects of the current world state. Contingent planning can be transformed into an AND/OR search problem in belief space (the space whose elements are sets of possible worlds). The plan, which is a tree rather than a sequence of actions, can treat every possible outcome. It means that the search space is an AND/OR tree, and the plan is sub-tree where all leaves are goal (belief) states. The main advantage of contingent planner is minimizing time to find the target, and execute manipulation tasks efficiently (because there is a plan with observation) (Hoffmann and Brafman, 2005). On the other hand, conformant planning is the special case of contingent planning where no observations are possible (Brafman and Hoffmann, 2004), (Koenig et al., 2004). According to different applications, approaches and strategies may vary. Some researches have used FF planners for manipulation tasks, like (Srivastava et al., 2014), (Akbari et al., 2018a), (Lagriffoul et al., 2013).

Besides the aforementioned approaches, some knowledge-driven approaches use ontologies in planning. For instance, a knowledge-based task and motion planning framework based on a version of the FF task planner is presented in (Akbari et al., 2016c). A reasoning process on symbolic literals in terms of knowledge and geometric information about the workspace, together with the use of a physics-based motion planner are used to evaluate the applicability and feasibility of manipulation actions and to compute the heuristic values that guide the search. The manipulation problem, involving knowledge about the world and the planning phase, is coded in the form of an ontology, and addresses a high-level and a low-level reasoning processes to appraise manipulation actions and prune the task planning phase from dispensable actions (Akbari et al., 2018b). Moreover, in (Beßler et al., 2018a) an assembly-based planner presented in Ontology Web Language (OWL) format using Description Logic (DL) is built upon an existing planner to assemble a kid toy plane called "BATTAT". This approach extends the proposed planner with a notion of action, and geometric reasoning capabilities. Actions are represented in terms of the action ontology which also defines action pre-conditions. Pre-conditions are ensured by running the planner for the action entity. This is used to ensure that the robot can reach an object, or else tries to put away occluding objects. To this end a geometric reasoner is integrated with the knowledge base. The interfaces of the geometric reasoner are hooked into the logic-based reasoning through procedural attachments in the knowledge base. Since this planner has been used in this thesis, we introduced in detail in Sec. 2.7.

2.1.2 Motion planning

Motion planning deals with detecting collision-free paths to convey a robot to a configuration state. It is mostly done in the configuration space ($\mathcal{C}$-space) (Lozano-Perez, 1983). The $\mathcal{C}$-space has as many dimensions as degrees of freedom the robot has, and therefore each point represents a configuration of the robot. The subspace corresponding to collision-free configurations is called $\mathcal{C}_{free}$ and the subspace corresponding to collision configurations is called C_{obs}. Motion

planning in $\mathcal{C}$-space consists in finding a path in $\mathcal{C}_{free}$ between two configurations.

With regard to classical motion planning, researches have investigated how to deal with geometric constraints in planning (they do not impose differential dynamic constraints). Traditional methods comprise different techniques such as cell decomposition, roadmap-based methods, and potential fields. In motion planning based on cell decomposition techniques, the first step is to decompose free space, either in an exact or an approximated way, and represent the set of cells as a graph. Then, some search algorithms such as *A** (Hart et al., 1968) or *Dijkstra* (Dijkstra, 1959) methods can be employed to find a path of minimal cost. The roadmap-based approaches build a graph to connect the initial and goal states in free space and they usually find the solution paths based on the shortest distance. Regarding this case, when the obstacles in the $\mathcal{C}$-space are represented as polygons, the visibility graph methods (De Berg et al., 2000) generates a graph whose nodes are the vertices of the polygons and whose edges are line segments with no interference of obstacles. The last approach uses potential fields computed in the $\mathcal{C}$-space to provide attraction to the goal configuration and repulsion from obstacles (Khatib, 1986). The drawback of all these approaches is that for problems with more than two or three degrees of freedom, they are hard or impossible to implement and computationally prohibitive because they require the construction of the $\mathcal{C}_{obstacle}$ in the configuration space.

Recently, much study is centered in sampling-based motion planning to provide efficient solutions for path planning by avoiding the need to compute the whole $\mathcal{C}$-space. It results in probabilistic complete planning. The core of sampling-based motion planning is to use random-based techniques to sample the $\mathcal{C}$-space and to interconnect those collision-free configurations as roadmaps or trees to capture the connectivity of $\mathcal{C}_{free}$. To cope with high dimensional degrees of freedom, it is found as an appropriate strategy to reduce the complexity of problems. Some sampling-based motion planners are investigated by (Elbanhawi and Simic, 2014). The most popular approaches are tree-based techniques, which provide a single query, and roadmap-based techniques, which provide multiple queries as described below.

If planning is carried out in static environments that remain constant at each manipulation step, roadmap-based approaches provide better solutions (even though they are costly to be implemented due to the need of exploring the whole $\mathcal{C}$-space) because they make it possible to set multiple queries, like Path planning based on the *Probabilistic Roadmap Method (PRM)* (Kavraki et al., 1996) that works in two phases. The first phase is the construction phase, that spends a specific amount of time sampling $\mathcal{C}_{free}$ and interconnecting samples with simple collision-free paths forming a roadmap. The second phase is the query process, which connects a start configuration to a goal configuration by using graph search techniques.

In manipulation planning, however, since the status of the environment is altered after executing each action, tree-based algorithms have better solutions because they provide specific explorations of the $\mathcal{C}$-space, e.g., focused on a given query. In this case, the environment is reconfigured and the $\mathcal{C}$-space must be again explored. Here, some of the well-known single query sampling-based motion planners are reviewed as they fit better in manipulation problems and some of them will be used later in this thesis.

Path planning based on the *Rapidly Exploring Random Tree (RRT)* (LaValle and Kuffner, 2001) explores the configuration space by expanding several branches of a tree. In the generic RRT algorithm, a tree is initialized at the root where the initial state is placed and it incrementally grows towards the goal configuration along random directions biased by the less explored areas. The *RRT-Connect* planner (Kuffner and LaValle, 2000) is a variant of *RRT*. There are two trees rooted at the start and goal configurations that grow to meet each other. The method provides substantial improvements in the search efficiency. The *RRT** planner (Karaman and Frazzoli, 2011) has been developed that minimizes the cost of the returned solution. After growing the tree as done in the basic *RRT* algorithm, the *RRT** captures the set neighbor nodes $\mathcal{N}$ of each new added nodes to verify if it can be reached through them with a less cost and, in this case, edges are rewired. Then, this procedure is repeated for the nodes in $\mathcal{N}$, which may to be rewired accordingly.

Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE) planner is particularly designed for complex dynamical systems (Şucan and Kavraki, 2009). *KPIECE* grows a tree of motions by applying randomly sampled controls for a randomly sampled time duration from a tree node selected as follows. The state space is projected onto a lower dimensional space that is partitioned into cells in order to estimate the coverage. As a result of this projection, each motion will be part of a cell, being each cell classified as an interior or exterior cell depending on whether the neighboring cells are occupied or not. Then, the selection of the cell is performance based on the importance parameter that is computed based on: 1) the coverage (the cells that are less covered are preferred over the others); 2) the selection (the cells that have been selected less number of time are preferred); 3) the neighbors (the cells that have less neighbors are preferred); 4) the selection time (recently selected cells are preferred); 5) the expansion (easily expanded cells are preferred over the cells that expand slowly). The cell that has maximum importance will be chosen, and a node of one of the motions of the cell will be randomly selected. The process continues until the tree of motions reaches the goal region. This approach is used while using physics-based motion planning, which is also used for grasp-in-the-clutter tasks, where the robot can interact with the obstacles obstructing the path towards the goal, moving them away.

2.1.3 Combination of task and motion planning

The increasing emphasis on real world applications has led AI planning researchers to develop algorithms and system that more closely match realistic planning problems, in which the required planning activity is often distributed or continual. Distributed planning refers to situations where the planning activity is distributed across multiple robots, processes or sites. Continual planning refers to an ongoing, dynamic process in which planning and execution are interleaved. A set of actions should be considered in each task description. For instance, for a manipulation task, the set of actions may include transit (a robot moves without having an object), transmit (a robot moves an object to the goal), push, pull, pick and place. These actions should be organized in terms of the constraints that may exist between them, and an efficient strategy is required to select the sequence of excutable actions. For applying each action, a task

planner has to evaluate whether the preconditions are satisfied or not in order to perform the required action. Complementary to motion and manipulation actions, sensing and reasoning actions can also be defined and used to reduce uncertainty and find a robust plan.

In general, combination of task and motion planning needs to search in symbolic and geometric space. A task plan, including a sequence of symbolic actions, does not promise that actions become feasible in terms of geometry conditions. This requires the combined search and motivates the researchers to investigate efficient ways of the combination at both levels in order to acquire a feasible plan. Combining task and motion planning is an active research area in *Robotics* and *AI*. This section explains different strategies employed to combine task and motion planning with respect to different types of applications where the initial state and action effects are fully observable.

There are two main ways of combining task and motion planning information: *simultaneously* or *interleaved*. The *simultaneously* approach like the work presented in (Akbari, 2018), accounts for geometric information by calling a motion planner while task planning is being pursued. The manipulation plan is then available after the task planning process is terminated. The *interleaved* approach like the work presented in Srivastava et al. (2014), decouples motion planning from the task planning part. Task planning first generates the sequence of actions, and then call a motion planner to evaluate the feasibility of the plan. Upon failure, geometric constraints are fed back to the task planner and the process resumes. This can be repeated several times until a feasible manipulation plan is achieved.

Next, the *TAMP* approaches are presented in which task planning and geometric reasoning are more tightly intertwined based on the proposed ways of combination.

FF-based TAMP: In this line, these studies are based on the *simultaneously* combination method. The studies in (Cambon et al., 2009a) present an algorithm which searches at symbolic and geometric levels, where a motion planner calls the task planner to guide roadmap sampling. Guidance is provided by a heuristic value based on the symbolic distance to the goal. In (Garrett et al., 2015) proposed an approach, called *FFRob*, which computes the heuristic value by analyzing the feasibility of actions with a *Conditional Reachability Graph (CRG)* based on a modification of *PRM* planner. It requires a pre-processing step to initialize the *CRG* by sampling objects poses and robot configurations, and determining conditions under which these samples are reachable or not.

In (Akbari, 2018), a simultaneous TAMP approach is proposed to efficiently deal with bi-manual robot manipulation problems in constrained environments. The proposed approach is a heuristic-based planner, which searches for a plan in state space, and takes into consideration geometric constraints while computing heuristic values. Specifically, two types of geometric reasoning processes are used: a) geometric reasoning about placements of objects, grasping poses, and inverse kinematic solutions; b) geometric reasoning about motion. The former involves Spatial, Reachability, and Manipulation reasoning, which are used to account for geometric constraints in heuristic values. The proposed heuristic is able to cover a large range of tabletop manipulation problems. It figures out and excludes unfeasible motion planning queries

which have kinematic problems or collisions in their start and goal configurations, and guides state space search. The latter calls a motion planner for validating state transitions and either returns a path or provides feedback in case of failure. The state of planner is updated by the constraints which are detected using both geometric reasoning processes.

Hierarchical-based TAMP: The works in this direction follow the *simultaneously* strategy. The work in (de Silva et al., 2013) focuses on a combination based on the *HTN* planner. It facilitates backtracking at different levels, also including an interleaved backtracking procedure. The application of combining hierarchical task and motion planning has been used by (Alami et al., 2014) and (Alili et al., 2010) for a teammate robot and robotic assistant. They use different geometric reasoning processes to find out a feasible plan.

LTL-based TAMP: LTL-based motion planning is a hybrid approach that provides a framework to describe complex motion planning tasks in terms of temporal goals, and that plans in discrete and continuous spaces. The work in (He et al., 2015) applied *TAMP* using the *LTL* task planner. This approach is done following the *interleaved* approach. Motion planning evaluation launches after a task plan is provided. In the case of failure, task planning input can be updated by a set of constraints in order to find another plan. The authors claim the planner is capable enough in moving away objects that block desired executions without requiring backtracking.

Constraint-based TAMP: In this direction, all the approaches are based on the *interleaved* combination method. The work in (Lagriffoul et al., 2012) and (Lagriffoul et al., 2014) introduce the concept of *geometric backtracking*, which denotes the systematic search process in the space of grasps and placements when instantiating a symbolic plan. They use linear constraints generated from symbolic actions and the kinematic model of the robot in order to prune the space of grasps and placements. The work in (Dantam et al., 2016a) proposed the *Iteratively Deepened Task and Motion Planning* method using the *Satisfiability Modulo Theories (SMT)*. It incrementally detects constraints and keeps dynamically adding or eliminating a number of task constrains based on the feedback obtained from the *RRT-Connect* motion planner. The approach is able to find an alternative plan when an unfeasible one is identified. It first finds the task plan, and then motion planning is employed to evaluate its feasibility. The work presented by (Lagriffoul and Andres, 2016) addresses *TAMP* by solving a culprit detection problem. In the case of failure at the geometric level, a logical explanation is computed. This explanation is fed back to the *Answer Set Programming (ASP)* task planner, which prunes entire families of plans leading to similar failures. The cycle repeats until a feasible plan is found.

2.2 Knowledge representation using ontologies

Knowledge representation techniques are concerned with structuring conceptual knowledge such that it is usable for reasoning tasks done by artificial systems (e.g. robots). A brief introduction to ontologies and its languages is the following:

Ontology: An ontology is a structured way of representing knowledge. Formally, it is defined as "an explicit, formal specification of a shared conceptualization" (Gruber, 1995). In this definition, the term of conceptualization refers to the abstract models of an entity in a domain. These models are achieved by defining its relevant concepts along with their relations. The simplest example of an ontology is a taxonomy. It is a hierarchical structure expressing the universe of discourse based on relations, such as *is-a* and *has-a* between concepts and instances of a class. These concepts, instances and relations are expressed in formal languages.

Formal languages for representing knowledge: Formal languages are used to express ontologies (in terms of concepts, relations and instances) and directly affects the reusability of the ontologies. There are three main categories of formal languages for representing knowledge: Frame based, First-Order Logic based and Description Logic (DL) based languages (Baader et al., 2017).

1. *Frame based:* It is a primary data structure used in artificial intelligence. Frame based language is originally developed for deductive databases. It focuses on explicit and intuitive representation of knowledge. The languages that are derived from namely frame based are: F-logic, Open Knowledge Base Connectivity (OKBC), and Knowledge Machine (KM).
2. *First-Order Logic:* It is a standard for the formalization of mathematics into axioms (facts). It allows the use of sentences that contain variables. The languages that are derived from First-Order Logic are: Common Logic, CycL, and Knowledge Interchange Format (KIF).
3. *Description logic (DL):* It provides an extension of namely frame based, without going so far as to first-order logic. The languages that are derived from DL are: KL-ONE, RACER, and OWL. A more detailed description of ontologies and DL can be found in Appendix B

Abundant studies investigated the use of robot knowledge in order to connect low-level data (sensors) with high-level information (ontology). These studies cover several areas such as object recognition, task planning, or navigation. Recently, some ontologies standards in robotics have been established by integrating some of the works done in these areas, with the aim of making knowledge more general, and shareable across many domains. Ontologies standards will be explained in subsection 2.4.

2.3 The use of knowledge in different domains

Many studies have investigated the use of knowledge in planning, like (Tenorth and Beetz, 2009) and (Ruiz-Sarmiento et al., 2017), that categorize knowledge about the world into terminological knowledge (TBOX), and assertional knowledge (ABOX). The former contains a hierarchy of concepts, such as interaction and action, and their relations, whereas the latter contains individuals that are instantiations of these concepts. In the navigation area, some

works such as (Lim et al., 2011), (Gemignani et al., 2016) and (Krieg-Brückner et al., 2005) use a metric map and a topological map to define the robot environment. The metric map is used for the geometrical representation of the robot workspace in terms of free and occupied areas, while the topological map is used to capture the topology of the workspace. In the manipulation planning domain, works such as (Akbari et al., 2018c) propose an ontological framework to organize the knowledge needed for physics-based manipulation planning, allowing to derive manipulation regions and behaviors. Other studies have investigated the use of the robot knowledge in specific fields in order to connect the existing low-level data with the high-level information (Rai and Hong, 2013) and (Chang et al., 2009). This connection can be used in several directions: object recognition and categorization (Johnston et al., 2008a), context modeling (Young Cheol Go and Joo-Chan Sohn, 2005) and context reasoning (Anagnostopoulos and Hadjiefthymiades, 2009), (Dargie, 2009), task planning (Cambon et al., 2009b), locomotion (Chatterjee et al., 2005).

For object recognition and categorization, a visual concept ontology that is composed of spatial concepts, spatial relations, color concepts, and texture concepts can all be used as an intermediate layer between domain knowledge and image processing procedures during the knowledge acquisition phase. Algorithms have also been developed for visual concept learning, feature selection, and training (Maillot et al., 2004). Moreover, symbol grounding, which occurs between sensory features and their symbolic representations (Johnston et al., 2008b), requires robust object recognition methods, combining many local features with several other visual features such as shape, color, and texture in a probabilistic and/or ontological approach.

2.4 Knowledge upper-level foundations efforts

Nowadays, the use of human-robot or robot-robot collaboration is playing a significant role in robotics. One of the basic requirements for any type of collaboration is the need for a common vocabulary along with clear and concise definitions. Therefore, the need for a standard and well-defined knowledge representation is becoming more evident. The standards of ontologies are discussed in (IEEE-SA, 2015). Ontologies for Robotics and Automation Working Group (ORA WG) is one of the recommended standard in robotics (Schlenoff et al., 2012), as explained below.

ORA WG: The goal of this working group is to develop a standard ontology and associated methodology for knowledge representation and reasoning in robotics and automation, together with the representation of concepts in an initial set of application domains. The standard provides a unified way of representing knowledge. It allows to transfer knowledge among any group of humans, robots and other artificial systems. This group is comprised of four sub-groups entitled: Upper Ontology/Methodology(UpOM), Autonomous Robots (AuR), Service Robots (SeR), and Industrial Robots (InR).

There are several general robotics standards that have been considered in the

standardization of service robots:

1. ISO 8373:1994, ISO 9787:1999, ISO 11593:1996, ISO 14539:2000, for industrial robotics. (under revision to include Service Robots)
2. ASTM E2521-07, for Urban Search and Rescue Robotic Operations
3. AIAA: S-066-1995, for Space Automation and Robotics
4. AJIS B 0144:2000, JIS B 0185:2002, JIS B 0186:2003, JIS B 0187:2005, TR B 0010:1999, for Assembly, Intelligent, Mobile, Service and Personal Robots.

The working group on ontologies for robotics and automation (ORA WG), sponsored by the IEEE Robotics & Automation Society, proposed standardized ways of representing knowledge for service, industrial, and autonomous robots (Schlenoff et al., 2012). ORA WG has proposed the Core Ontology for Robotics and Automation (CORA) (Prestes et al., 2013), a conceptual structure to be used and integrated within other specific ontologies developed for the robotics and automation domain, i.e., with a main focus on reusability.

The structure of CORA is based on the Suggested Upper Merged Ontology (SUMO) (Niles and Pease, 2001b), which is a top-level ontology that aims to define the main ontological categories describing the world. Later, in order to provide the more generalized concepts of the robotic domain, CORA was extended by incorporating CORAX (Fiorini et al., 2015), RPARTS (Robot parts), and POS (Position) (IEEE-SA, 2015). CORAX is an ontology that defines the concepts not explicitly or completely covered by SUMO and CORA, such as robot motion (that categorizes the motion of the robot according to its type such as robot rolling or walking). RPARTS provides a general information of robot parts, also defining the attached parts on the robot such as robot sensing parts. POS captures general information about position and orientation. More recently, CORA has been enhanced by adding the knowledge, called Autonomous Robot Architecture Ontology (ORA), which defines the main concepts and relations regarding robot architecture for autonomous robots (Olszewska et al., 2017). The IEEE 1872 standard covers all these ontologies and provides them in OWL form (https://github.com/srfiorini/IEEE1872-owl).

Besides SUMO and CORA, an ontological foundation for structural conceptual models called UFO (Guizzardi, 2005) provides high-level concepts that are linked to those used in the domain specific ontologies. It focuses on structural modeling aspects, this foundational ontology is an ontology of objects, their properties and relations, their parts, the roles they play, and the types they instantiate.

DOLCE UltraLite (DUL) (Masolo et al., 2003) makes a variety of important distinctions that are useful for upper ontologies. It is developed in a principled way to help ensure correct and consistent distinctions. The names of classes and relations in DUL have been made more intuitive. Moreover, temporal and spatial relations are simplified. Axiomatization makes use of simple constructions.

Nowadays, the Everyday Activity Science and Engineering (EASE) [1] ontology has used DUL ontology and its concepts are inherited from it, which was chosen because of it has more cleaner ontological model focusing on the cognitive categories humans use to describe the world. EASE is the study of the design, realization, and analysis of information processing models that enable robotic agents (and humans) to master complex human-scale manipulation tasks that are mundane and routine.

2.5 Knowledge-based semantic perception in robotics

The semantic perception is devoted to object detection, object recognition, segmentation and scene understanding. One of the first problems in semantic perception deals with labelling objects or some parts of an environment for different purposes, such as human-robot or multi-robots collaboration. Then, the semantic gap problem must be solved, i.e. to address the difference between the formulation of an entity represented in the form of high level (qualitative) knowledge and its low level (quantitative) computational representation (e.g., sensor data). For instance, in the domain of computer vision and information retrieval, the semantic gap refers to the gap between extractable information from (visual) data, and the interpretation that a user has about the data. The semantic gap problem is tackled in the design of knowledge bases, which in general need to be closely integrated with the sensing and the acting. Some of the software tools developed for this purpose are the following:

1. *DyKnow:* The knowledge processing middle-ware introduced in (Heintz et al., 2010) provides a generic software support for modelling and maintaining objects, states and events of complex environments for the purpose of traffic monitoring. The processes in this middle-ware are done at different levels of abstraction from numeric sensor data to qualitative information grounded in the world. The abstract model provided by DyKnow, in conjunction with the reasoning methods, enable the system to close the (sense-reasoning) gap between numeric data and high level knowledge.

2. *KnowRob:* Introduced in (Tenorth and Beetz, 2017) it is an ontological knowledge processing system designed for autonomous robots. The main objective of this knowledge processing model is bridging the gap between the high-level task descriptions given to the robot and the detailed information gathered from external sources such as the perception system of the robot. For this to be possible, KowRob is equipped with a knowledge base which creates a symbolic layer upon the perceptual information of the robot. Containing high-level knowledge about the domain and the robot, the symbolic layer enables logical inference over both the perceptions and the internal data about the control configuration of the robot.

3. *K-COPMAN (Knowledge-enabled Cognitive Perception for Manipulation):* Introduced in (Pangercic et al., 2010), this system is also designed to provide the abstract symbolic

[1] https://github.com/ease-crc/ease_ontology

knowledge for the perception of the robot. The combination of the symbolic knowledge in aforementioned knowledge models with the low-level data enables the robot to reason about its perceptions, which can enable the robot to make better decisions during its task execution.

These approaches are used in specific tasks, and huge efforts are required to adapt them to new scenarios. This means no common vocabularies or a common structural way (standardized way) are available when modeling a new task. This issue decreases the interoperability of using those frameworks in different tasks.

2.6 Use of ontologies to increase robot autonomy

The knowledge-based approach has been used to increase the robot autonomy. The fields of study mentioned above have thrived independently. Along their advances, some of them have been combined together for robotics applications. The TAMP framework is an example of successful combinations. High-performance perception is a primary capability of autonomous robots, so it has been used by many robots across different domains. On the other hand, there has been a line of research on architecture designs that use knowledge formalism and reasoning capabilities to fully integrate the components necessary for autonomous execution of tasks.

The system architecture proposed in (Keleştemur et al., 2019) supports robot manipulation services. Like (Shah et al., 2019), the architecture aims to perform tasks in various scenarios of the RoboCup@Home league. The authors point out that the majority of research in manipulation focuses on solving each of the problems of perception, grasping, motion planning, and user interfaces. Some works propose integrated systems but they still need human intervention (Jain and Kemp, 2010) or a hard coded 3D map of objects (Srinivasa et al., 2010). On the other hand, the architecture proposed in (Keleştemur et al., 2019) aims to develop a complete system including the user interface through natural language. The architecture is tested in real-world environments using a mobile manipulator in the RoboCup league and show competitive results. However, the architecture lacks the ability to store knowledge and reason with it. The method used for task planning is not discussed and seems to be implemented simply as finite-state machines. Also, the test scenario introduced is relatively simple as it does not consider cluttered environments, different approaching angles for grasping, navigation of the mobile base, and searching where to place picked objects.

In (Lim et al., 2011), low-level sensory data has been integrated with high-level knowledge (context information) for robot intelligence. The authors describe a unified robot knowledge within a concept hierarchy through the use of an ontology to increase the robot autonomy and facilitate the navigation process. In (Galindo et al., 2008), a semantic knowledge is proposed to be profitably used for robot task planning. A specific type of semantic maps, which integrate hierarchical spatial information and semantic knowledge is proposed. These semantic maps can improve task planning in two ways: extending the capabilities of the planner by reasoning

about semantic information, and improving the planning efficiency in large domains. The effectiveness of these semantic maps is demonstrated in a domain involving robot navigation in a domestic environment. In (Kunze and Beetz, 2017), a framework that allows to envision the physical effects of robot manipulation actions is proposed. The envisioning is considered to be a qualitative reasoning method that reasons about actions and their effects based on simulation-based projections. Thereby it allows a robot to infer what could happen when it performs a task in a certain way. This is achieved by translating a qualitative physics problem into a parameterized simulation problem; performing a detailed physics-based simulation of a robot plan; logging the state evolution into appropriate data structures; and then translating these sub-symbolic data structures into interval-based first-order symbolic, qualitative representations, called timelines. The result of the envisioning is a set of detailed narratives represented by timelines which are then used to infer answers to qualitative reasoning problems. In (Nam et al., 2020) a software architecture is proposed that integrates the components necessary to perform tasks by a real robot without human intervention. The architecture is composed of deep learning-based perception, symbolic reasoning, AI task planning, and geometric motion planning. A deep neural network is implemented that produces information about the environment, which is then stored in a knowledge base. Moreover, a reasoner processes that knowledge and uses the result for task planning. A combination of task and motion planning is used through an proposed interface that computes geometric information necessary for motion planning to execute the symbolic task plans. The architecture is developed based on Robot Operating System (ROS) so compatible with any robot that is capable of object manipulation and mobile navigation running in ROS.

Based on the review of existing approaches, we find that there exists a substantial gap between the potential of robot autonomy and the reality and the following concerns should be addressed. First, existing architectures, often do not have all functionalities for perception, knowledge representation, reasoning, task and motion planning, and user interface. They lack more than one of the functionalities. Second, many of the existing architectures require some level of human intervention to complete the given tasks. Another concern is the underrepresented importance of demonstrations with physical robots for nontrivial tasks in real-world environments. Moreover, most of the previous works do not show physical robot experiments across different robot hardware platforms although they claim that their architectures are not tied to particular platforms. Therefore, the main contribution is to increase the robot autonomy by integrating perception, knowledge engineering, reasoning, and task and motion planning/repair the plan or re-planning, which is deployed to different robot platforms to plan and perform for object manipulation.

2.7 Logic-based planning

The logic-based planning (or knowledge-driven planning) is a planning approach that has recently been used in the manipulation domain, which provides fixed recipes to be done by the robot. This approach has been introduced in (Beßler et al., 2018a), (Kootbally

et al., 2015), (Balakirsky et al., 2013), (Imran and Young, 2015) for manipulation assembly problems. Since the work presented in (Beßler et al., 2018a) has been used in this thesis, a brief description of it will be introduced. In this work, the recipes are elegantly represented using description logic (DL). The sequence of assembly actions have been inferred using a reasoning mechanism to be performed in the current situation. The reasoning mechanism infers information over several ontologies like, assembly ontology to teach the robot how to assemble two parts, action ontology to let the robot reason over the best action in the current situation, and the planning ontology which provides the agenda of how to execute the task. They are detailed below.

Assembly Ontology

The upper level of the assembly ontology defines general concepts such as *MechanicalPart* and *AssemblyConnection*. Underneath this level there are part ontologies that describe properties of parts such as what connections they may have, and what ways they can be grasped. Finally, assemblage ontologies describe what parts and connections may form an assemblage. This layered organization allows part ontologies to be reused for different assemblies. Also important is the *Affordance* concept. Mechanical parts provide affordances, which are required by grasps and connections. Apart from these very abstract concepts, some common types of affordances and connections are also defined (e.g. screwing, sliding, and snapping connections).

Action Ontology

At some point during the planning process, the robot has to move its body to perform an action. Explicit action representations are used to ensure that pre-conditions are met before performing an action. The action ontology includes relations to describe objects involved, sub-actions, etc.

The modelling of action pre-conditions is based on the *preActors* relation which is used to assert axioms about entities involved in the action that must hold before performing it. The upper ontology also defines more specific cases of this relation such as *objectActedOn* that denotes objects that are manipulated, or *toolUsed* that denotes tools which are operated by the robot.

ConnectingParts The most essential action the robot has to perform during an assembly task is to connect parts with each other. At least one of the parts must be held by the robot and moved in a way that establishes the connection.

The relations *assemblesPart* $\sqsubseteq$ *objectActedOn*, and *fixedPart* and *mobilePart* $\sqsubseteq$ *assemblesPart* are defined. These denote *MechanicalPart*'s involved in *ConnectingParts* actions, and distinguish between mobile and static parts. Further, the relation *assemblesConnection* is defined that

denotes the *AssemblyConnection* the action tries to establish. The *assemblesPart* relation is defined as property chain *assemblesConnection* ∘ *hasAtomicPart*, where *hasAtomicPart* denotes the parts linked in an *AssemblyConnection*. This ensures that *assemblesPart* only denotes parts that are required by the connection. Using these relations the following axioms are asserted for the *ConnectingParts* action:

$$\leq 1 assemblesConnection.Thing \wedge \geq 1 assemblesConnection.Thing \tag{2.1}$$

$$\geq 2 assemblesPart.Thing \tag{2.2}$$

$$\leq 2 mobilePart.Thing \wedge \geq 1 mobilePart.Thing \tag{2.3}$$

These axioms define that: (2.1) an action is performed for exactly one assembly connection; (2.2) at least two parts are involved; and (2.3) at max two parts are mobile, and at least one mobile part is involved.

Another pre-condition is the grasp ability of mobile parts. Parts may relate to *GraspingAffordance*'s that describe how the robot should position its gripper, how much force to apply, etc. to grasp the part. Next, a property *partConnectedTo* is defined that relates a part to parts it is connected to. It is sub-property of the property chain $hasAtomicPart^{-} \circ hasAtomicPart$. Also, this relation is transitive such that it holds for parts which are indirectly linked with each other. This is used to assert that fixed parts must be attached to some fixture:

$$\forall fixedPart.(\exists partConnectedTo.Fixture) \tag{2.4}$$

Also, parts must be in the correct fixture for the intended connection.

MovingPart and PutAwayPart The above statements assert axioms that must be ensured by the planner. These refer to entities in the world and may require certain actions to be performed to destroy or create relations between them, in order to ensure a valid spatial arrangement in the scene.

First, the robot should break non permanent connections in case one of the required affordances is blocked. This action is defined as $MovingPart \sqsubseteq PuttingSomethingSomewhere$. The only pre-actor is the part itself (it is linked to the action via the relation $movesPart \sqsubseteq objectActedOn$), and it must have an unblocked grasping affordance.

Further, parts that occlude required parts for an assembly step must be put away. We define this action as $PutAwayPart \sqsubseteq PuttingSomethingSomewhere$. This action needs exactly one *movesPart*, and additionally refers to the parts that should be "avoided", which means that the target position should not lie between the robot and avoided parts: $\exists avoidsPart.MechanicalPart$, where *avoidsPart* is another sub-property of *preActors*.

Planning Ontology

The planner is driven by comparing goals, represented in the TBox, with believes, represented in the ABox, and controlled by meta knowledge called planning strategy. The planning strategy determines which parts of the ontology are of interest in the current phase, how steps are ordered, and how they are performed in terms of how the knowledge base is to be manipulated. Possible planning decisions are represented in a data structure called planning agenda. Planning agendas are ordered sequences of steps that each, when performed, modify the belief state of the robot in some way. The planner succeeds if the belief state is a proper instantiation of the goal description.

Different tasks require different strategies that focus on different parts of the ontology, and that have specialized rules for processing the agenda. The strategy for planning an assemblage, for example, focuses on relations defined in the assembly ontology. Planning to put away parts, on the other hand, is mainly concerned with spatial relations. Strategies are associated to entities that should be planned with them. To this end, the relation *needsEntity* is defined that denotes entities that are planned by some strategy. Strategies assert a universal restriction on this relation in order to define what type of entities can be planned with them. For the assemblage planning strategy, for example, the axiom is asserted:

$$\forall needsEntity.(Assemblage \vee AssemblyConnection) \tag{2.5}$$

Planning decisions may not correspond to actions that the robot needs to perform to establish the decisions in its world. Some decisions are purely virtual, or only one missing piece in a set of missing information required to perform an action. The mapping of planning decisions to action entities is performed in a rule-base fashion. These rules are described using the *AgendaActionMapper* concept, and are linked to the strategy via the relation *usesActionMapper*. Each *AgendaActionMapper* further describes what types of planning decisions should activate it. This is done with agenda item patterns that activate a mapper in case a pattern matches the selected agenda item. These are linked to the *AgendaActionMapper* via the relation *mapsItem*.

Finally, the *AgendaActionPerformer* concept is defined which is linked to the strategy via the relation *usesActionPerformer*. *AgendaActionPerformer* provide facilities to perform actions by mapping them to data structures of the plan executive, and invoking an interface for action execution. They are activated based on whether they match a pattern provided for the last agenda item.

Although these ontologies are useful for such manipulation applications which have a well-structured environment, they lack to show how it can be used in semi/unstructured environments which requires the integration of a geometric reasoning module to check the feasibility of the actions in such cases of combining two parts or even if the path toward a certain object is a collision-free. Part of the current thesis will try to fill this gap.

Chapter 3

Knowledge Guidance for Task and Motion Planning

3.1 Introduction

This chapter introduces semantic manipulation knowledge for manipulation planning. The manipulation planning system includes the combination of task and motion planning (TAMP) levels. The use of abstract knowledge can enhance and facilitate the planning capabilities in both levels and give more autonomy to the robots to perform tasks. Many approaches exist for the representation of knowledge and one of them are ontologies, which are used to structure the knowledge in terms of concepts and relations. Reasoning processes over semantic knowledge can be then applied in order that robots infer the particular situation in their workspace.

3.2 Problem statement and proposed solution

3.2.1 Problem formalization

In manipulation planning, dynamic interactions between the objects and the robots play a significant role. In this scope, dynamic engines, such as Open Dynamic Engine (ODE-http://www.ode.org/) allow to consider them within motion planners, giving rise to physics-based motion planners that consider the purposeful manipulation of objects. In this context, on the one hand, at the geometric level, the representation of knowledge regarding how the objects have to be manipulated eases a semantic-based reasoning that reduces the computational cost of physics-based planners. In this work, an ontology framework is proposed to organize the

knowledge needed for physics-based manipulation planning, allowing to derive manipulation regions and behaviors. A semantic map is constructed to categorize and assign the manipulation constraints based on the robot, the objects and the type of actions. The ontology framework can be queried using Description Language to obtain the necessary knowledge for the robot to manipulate the objects in its environment.

On the other hand, at the symbolic level, autonomous indoor service robots are supposed to accomplish tasks, like serve a cup, which involve sequences of manipulation actions. Particularly, for complex manipulation tasks that are subject to geometric constraints, spatial information is required, together with the way in which these objects can be manipulated. In this line, in this chapter an ontological-based reasoning framework called Perception and Manipulation Knowledge (PMK) is proposed as a guidance module for task and motion planning. The PMK includes: (1) the modeling of the environment in a standardized way to provide common vocabularies for information exchange in human-robot or robot-robot collaboration, (2) a sensory module to perceive the objects in the environment and assert the ontological knowledge, (3) an evaluation-based analysis of the situation of the objects in the environment, in order to enhance the planning of manipulation tasks. The work describes the concepts and the implementation of PMK, and presents an example demonstrating the range of information the framework can provide for autonomous robots.

3.2.2 Proposed Framework overview

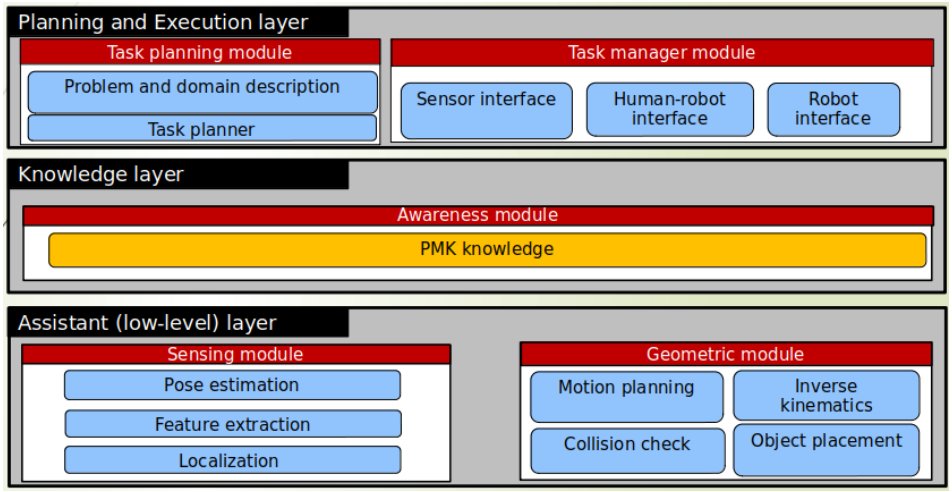


Figure 3.1: Knowledge-based reasoning framework for task and motion planning (KTAMP).

With the aim of executing tasks automatically, the integration of several layers and modules is required, covering perception, knowledge representation and reasoning, and planning at symbolic and geometric levels.

The proposed framework Knowledge-based reasoning for Task and Motion Planning (KTAMP) is composed of three main layers, as shown in Fig. 3.1 planning and execution, knowledge, and assistant (low-level) layer.

The planning and execution layer contains two modules, the task planning and the task manager modules. The former includes a task planner to compute a sequence of actions to be done, which requires a problem and domain description to set the initial scene, including the state of the world entities, and the goal state. The latter provides interfaces to communicate with the agents/operators (e.g., robots, humans or sensors). It also keeps monitoring the executed actions or atomic actions and it returns a failure signal to the recovery module if an error occurs. Moreover, it has a procedural structure for each step of a action. This structure is formally defined as a workflow that can be automatically executed through interfacing existing software components of the robot control system, e.g., executing a *pickUp* requires a call to the Inverse Kinematics (IK) module to check reachability for grasping the objects then finding a collision-free path towards a grasping configuration. A *putDown* action must search for available placement room.

The knowledge layer contains the Awareness module to guide the planning and execution layer. It is composed of the PMK knowledge that consists of manipulation and perceptual knowledge. The former assists the robot to figure out the environmental entities, workspace and planning. The latter assists the robot to figure out which are the proper algorithms and parameters to be used for the available sensors in order to extract data. The PMK knowledge is described in detail below in Sections 3.3 and 3.4.

Finally, the assistant layer provides the low-level modules that allow to deal with: perception issues, like finding out which sensors can be used for a sensing action in a given situation, which are dealt by the sensing module, and geometric issues, like determining if a configuration is collision-free or if an inverse kinematic solution exists for a gripper pose, which are dealt by the geometric module.

3.3 A Knowledge Processing Framework for Physics-based Manipulation Planning

Description of OUR-K Knowledge Classes

The ontology approach for physics-based manipulation planning proposed here is the first version of PMK. It is inspired by the Ontology-Based Unified Robot Knowledge for Service Robots in Indoor Environments OUR-K (Lim et al., 2011).

The main contribution of the OUR-K framework is the establishment of an ontology-based unified knowledge for service robots in human environments by integrating low-level data

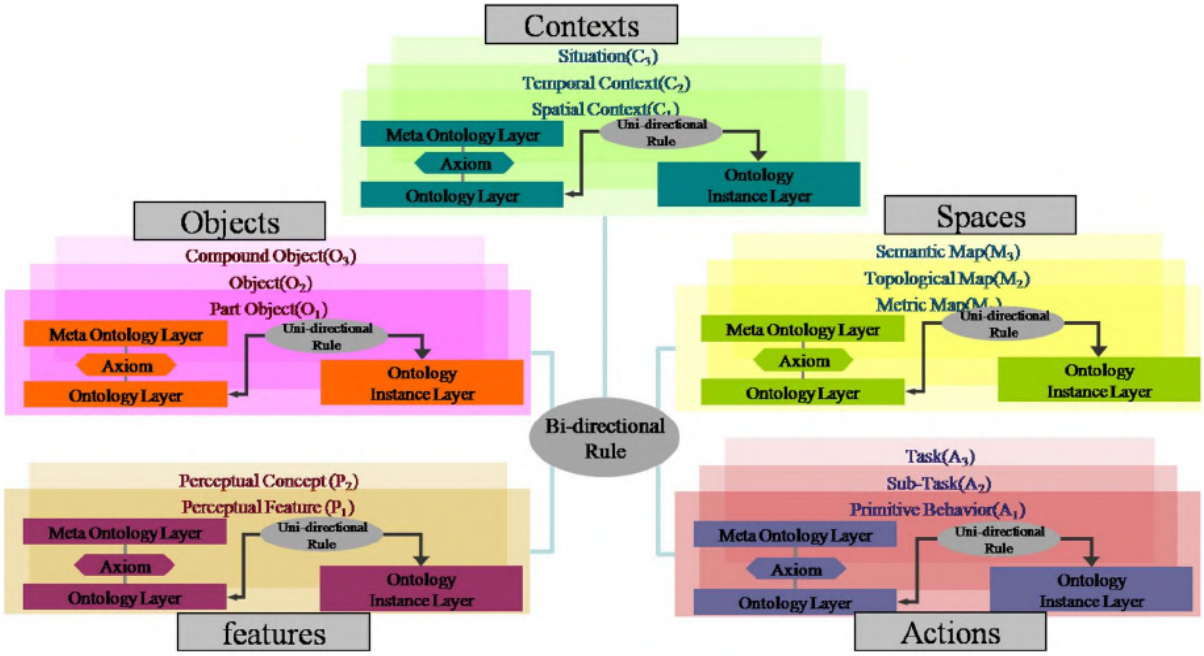


Figure 3.2: The structure of OUR-K ontology (Lim et al., 2011)

(sensory data) with high-level knowledge (context information). This framework is described within a concept hierarchy through the use of an ontology. As shown in Fig. 3.2, it is composed of five main classes: feature, object, space, context, and action. Each class has three levels (sub-classes) (except feature class that has two), and each level has three ontological layers: metaontology, ontology, and ontology instance. *Metaontology layer* is used to represent generic information, such as the concept of physical object in the service robotics field. *Ontology schema layer* is a layer used for domain specific knowledge, for instance, in service robotics, ontology layer contains the knowledge of a particular domain such as kitchen. *Ontology instance layer* is used to store the information of the objects (such as their features). Because ontologies in general are an object-oriented and frame-based language the metaontology layer can provide a template for ontology layer to build terminology, while the ontology instance layer can be defined as an individual frame. The information of ontological classes, properties, and instances is transferred within unidirectional reasoning in the same knowledge level. Whereas, bidirectional reasoning relates several knowledge classes or knowledge levels. OUR-K classes are described below:

Feature class is used to define how objects are perceived. It has two levels: perceptual feature and perceptual concept. In perceptual feature level, the real environment is described from the perceptual perspective, i.e. sensors, that perceive features of instances such as color, texture, and scale invariant feature transform (SIFT). In perceptual concept level, concepts are grounded to perceptual features.

Object class is used to define objects, their functionality and their parts. It is composed of three levels: part-object level, object level and compound level. Part-object includes parts of objects according to their functionality (for example, body and handle each has its own functionality, i.e. containing and grasping respectively). Object level includes object name and functionality, for instance, the composition of a *cup* is body and handle. In compound level, closely related objects that can be utilized together are linked (e.g. a *cup* and a *saucer*).

Workspace class builds a semantic map and uses it to enhance the representation of the environment, which requires geometrical, positional information and qualitative features, to express relations. The construction of a semantic map depends on two types of maps: metric and topological. The former defines areas in the environment, which may be empty or occupied. The latter contains the information of the topology of the free region (e.g. with a graph extracted from a voronoi diagram). The relation between these maps and objects is stored in the semantic map.

Context class is used to understand the situation of the objects in the environment. This situation is derived from two types of relations: spatial and temporal, which are defined in spatial and temporal levels, respectively, such as *crowd*, which means that the robot will encounter some obstacles. The spatial level contains the spatial relations (such as *on, in, left, and right* functions) and space classes (semantic map). Temporal level contains the temporal relations such as *before, after, overlap, meet*.

Action class is used to define a task and the way of execution it. Action class consists of three levels: primitive behavior, sub-task and task. In primitive behavior, perceptual, motion and manipulation behaviors are defined (e.g. *turn, goto, extractcolor, extractSIFT*). In a sub-task level, a short-term sequence of behaviors are defined, such as *gotoSpace, localization*. A task such as *navigate, delivery* is defined in task level. It can be decomposed into sub-tasks, which can be decomposed into primitive behaviors. For example, *recognizeObject* is a task of finding an object within an image (e.g. *extractcolor, extractSIFT*).

First Proposal of a Physics-based Manipulation Ontology

In previous works (Akbari et al., 2016a), the authors proposed a manipulation ontology to represent knowledge to face the manipulation problems a motion planner has to deal with when the robot moves and encounters obstacles in the environment. Two classification of objects are proposed in this ontology, fixed bodies and manipulatable bodies (i.e. obstacles that can be pushed away). The former remains static during the whole planning process, even if collision happens with other manipulatable objects. The latter can be manipulated during the planning. The manipulatable bodies are classified as free manipulatable bodies and constrained-oriented manipulatable bodies. The free manipulatable bodies can move in any direction (according to the dynamics of rigid bodies) when collision occur. The constraint-oriented manipulatable bodies have some allowable motion directions and others are restricted. The manipulation constraints are modeled by defining the manipulatable region from where the object can be

pushed from (i.e. where the robot should be located to apply a pushing force).

The knowledge is structured by defining six classes derived from a general manipulation class. These classes describe the initial state, goal state, action, region, object type, and object elements. Initial state class describes an initial location of the robot, while goal state class shows the final location of the objects. Action class contains different manipulation primitive actions, like pick, place and push. Region class defines three sub-classes: manipulation, object, and goal region. ManipulationRegion sub-class defines the region of manipulation, i.e. from where the mobile robot can interact with the object. ObjectRegion sub-class is defined as a bounding box of an object. GoalRegion sub-class is a circular region defined around the goal state. ObjectType class defines the manipulatable objects and free objects and their constraints. ObjectElements describe the feature of the objects.

Although this proposal covers the manipulation constraints, it lacks to cover knowledge related to environmental entities, perception devices, agents, and planning.

3.3.1 The Proposed Framework for Physics-based Manipulation

The representation of robot knowledge with an ontology has interesting properties like shareability and exchangeability (Alirezaie, 2015). Knowledge is represented by defining the properties and relations between concepts that provide a rich semantic description to aid the solving of particular problems. The aim of this work is to formalize a framework following the structure of the OUR-K ontology to cope with manipulation planning problems by providing the necessary knowledge (e.g, perception, and planning) for complete scenarios. As shown in Fig. 3.3, the proposed ontology framework has six classes. The object class and the context class are the same as in the OUR-K framework. The other four are described in the following subsections.

Feature class

In the proposed framework, the feature class has two levels: physical concept and physical feature. The concepts of $rigidbody$ and $interaction$ are defined in the concept level, and their features in terms of material and interaction parameters in feature level. Fig. 3.4 describes the ontology schema of the feature class. For example, interaction is a concept defined as *"Contact between two surfaces"*, while its features depends on the type of material, i.e. each material has its properties like friction coefficients, density, etc. They are linked together (material and its properties) via axioms (facts). For example, to handle a cup (which is made from A), the interaction occurs between two rigid bodies, cup and robot (its end-effector is made from B), and each has its physical properties, as described below using DL:

Interaction

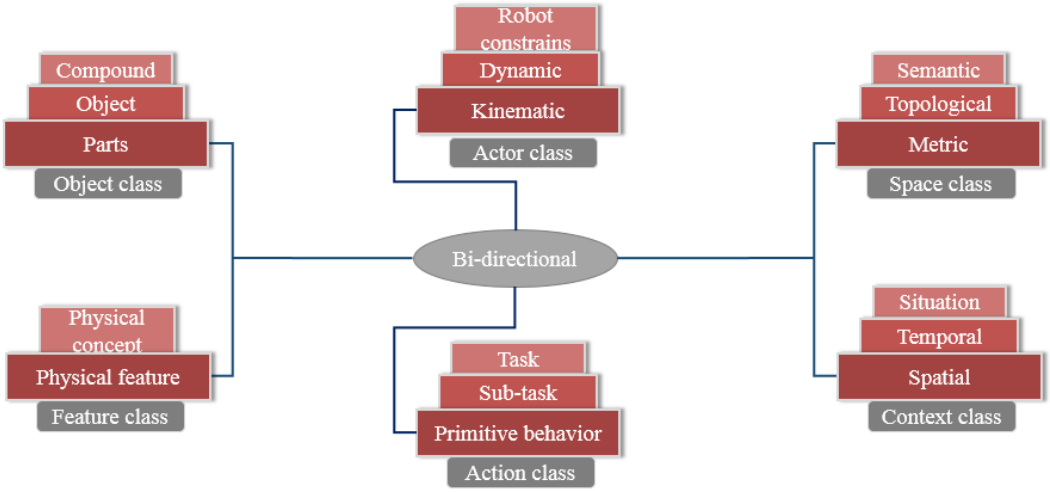


Figure 3.3: Classes of the proposed physics-based ontology for manipulation, based on the OUR-K framework.

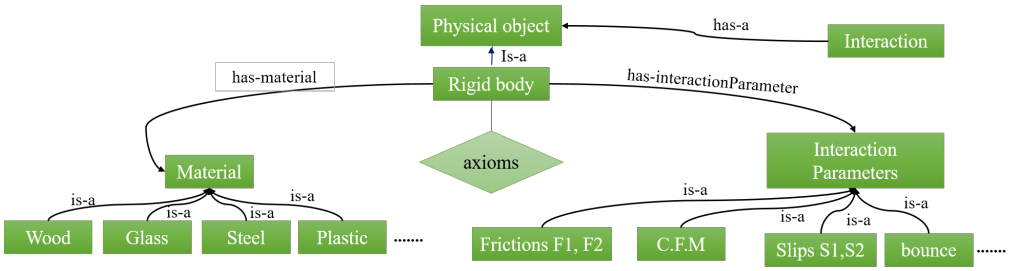


Figure 3.4: Schema of feature class that explains the hierarchy of the concepts, features, and relations via axioms

$$\wedge \exists hasSuperclass(Rigidbody, PhysicalObject)$$
$$\wedge \exists hasInterParameter(interactionParameter, Rigidbody)$$
$$\wedge \exists hasmaterial(material, Rigidbody)$$
$$\wedge \exists ismaterial(A, material)$$
$$\wedge \exists ismaterial(B, material)$$
$$\wedge \exists hasfeature(friction, interactionParameter)$$

Axioms of the feature class define that the rigid bodies have a constituent material and that this material has properties, such as friction coefficient, density, slip, CFM (Constraint Force Mixing), and bounce.

The physical features are used by the ODE simulator to accurately model the physical environment.

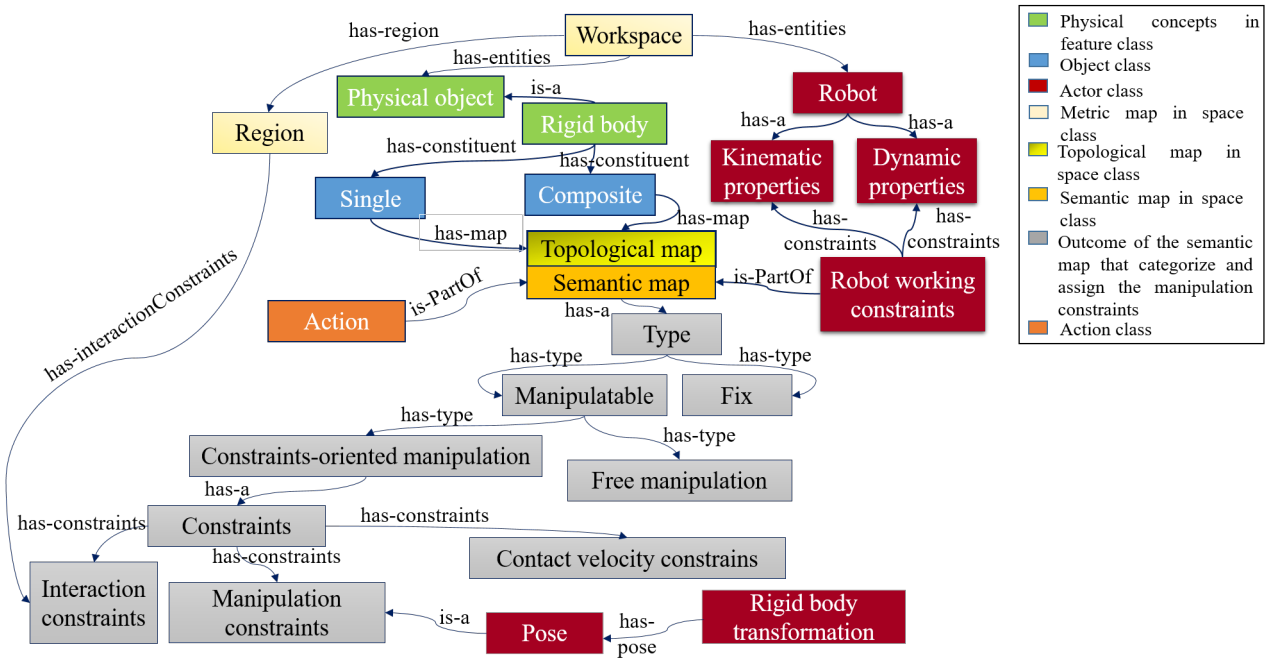


Figure 3.5: Schema of workspace class that describes the outcome of semantic map. The legend of different color shows that the classes information that is required to generate the semantic map.

Actor class

Actor class is a new-defined class that describes the properties of the robot in the environment. It consists of three levels: robot kinematics, dynamics, and constraints. Kinematic-level defines the kinematic structure of the robot and its location in the workspace, whereas dynamic-level contains the dynamic parameters of the joints of the robot, such as dumping, joint stiffness, and maximum efforts. In the constraints-level, based on kinematics and dynamics features, the robot working constraints are extracted.

Workspace class

In our framework, the Workspace class contains three knowledge levels: metric map, topological map, and semantic map. A metric map contains empty and occupied spaces (by either objects or the robots). Topological map defines the topology of the workspace, including where the objects are located in the workspace. Semantic map categorizes physical objects in the environment along the manipulation constraints. These constraints are defined as a function of the robot. It means that semantic map S depends on four parameters $S = (O, R, A, T)$, where: O is the set of objects that is provided by object class, R is the set of robot working constraints that is provided by the actor class, A is an action that is provided by the action class, and T is a topological

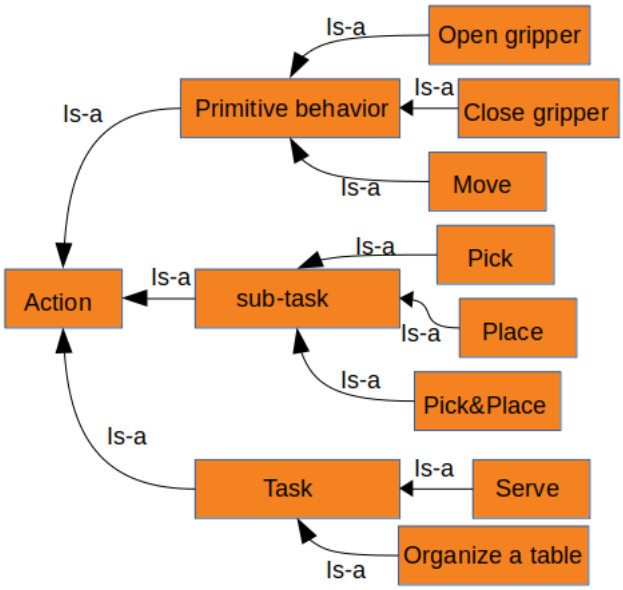


Figure 3.6: Schema of action class that is divided into three sub-classes: primitive behavior, sub-task and task.

map that is provided by the Workspace class. The outcome of the semantic map is to assign the manipulation constraint by using DL reasoning, i.e. as shown in Fig. 3.5, the outcome of the semantic map due to reasoning is the type of object and its constraints.

Action class

The action knowledge class is composed of three levels: primitive behavior, sub-task, and task levels. A task is decomposed into sub-tasks, while sub-tasks involve a combination of primitive behaviors. The ontology schema of the layers of the three levels in action class is instantiated by a planner. When requested to plan, the planner checks on the topological map to know which object (or robot) is occupied with which space. Then, the semantic map is used to extract the constraints of the object (single or composite), and the robot w.r.t the action. Context class, particularly temporal level, can be used by the planner to set the sequence of action of a task.

3.3.2 Usage in Manipulation Table-top Problem

To illustrate the proposal some simulation examples are performed using The Kautham Project (Rosell et al., 2014), which is a C++ based open-source tool for motion planning, that includes geometric, kinodynamic, and physics-based motion planners. It uses OMPL (Sucan et al., 2012) for the core set of planning algorithms. OMPL is a C++ based open-source library for sampling

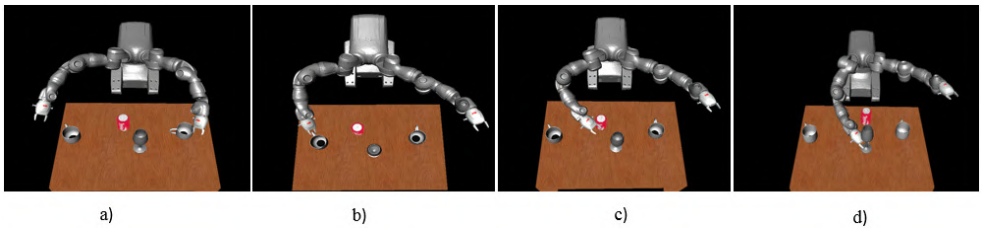


Figure 3.7: Kautham scenes that explain the scenarios of manipulation of the cup and wine-glass: a) push the cup, b) pick the cup, c) move the obstacle coca-can then, d) grasp the wine-glass.

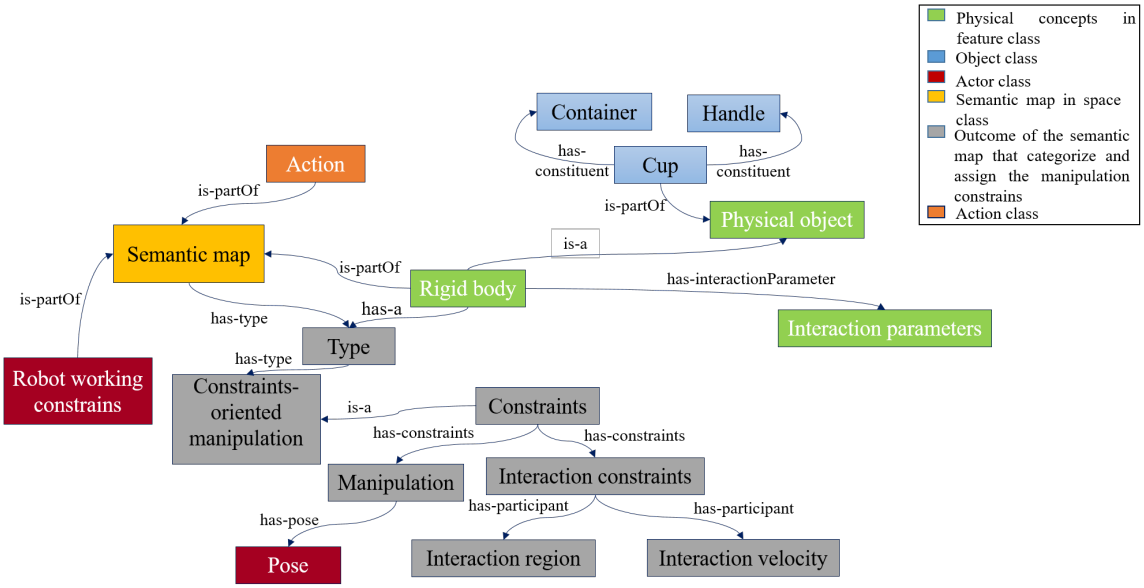


Figure 3.8: Example of the cup that shows the required information from classes to apply pick and push actions. The legend defines the type of classes.

based motion planning. Physics-based simulation are modeled using Open Dynamic Engine (ODE) that is used as state propagator for physics-based planning algorithms. Ontologies are encoded using the Web Ontology Language (OWL) (Antoniou and van Harmelen, 2004). XML-based file format is used to access and share such knowledge. The ontologies are designed using the Protégé editor (http://protege.stanford.edu/). It is an open-source software that provides an ontology editor to develop knowledge-based applications. The query over the ontological knowledge is performed using Prolog predicates using Swi-Prolog library.

A sequence of actions was computed using the task planning approach proposed in (Akbari et al., 2016c). In this work, a task and motion planning framework is proposed based on a modified version of the Fast-Forward (FF) task planner. The FF planner is coupled with physics-based motion planning to enable and take into account the interaction between rigid bodies, in order to determine in a realistic way how to execute the symbolic actions obtained by the task

planning level. In order to know the way of executing these actions, ontological knowledge can be used. A high level task such as $organizethetable$ can be satisfied by applying some actions like push, pick, place, grasp, etc. In the following examples, some queries to the ontology framework are presented using DL.

Fig. 3.7 shows a scene of The Kautham Project. It has two types of objects: cup (body and handle), that is a pickable or pushable object, and a Coca-can that is a pushable object. The aim is to show the manipulation behavior of the objects with respect to the constraints of the robot, the objects and the type of action. For example, the cup is a pickable object using its handle, or a pushable object using the body, as shown in example (1) and (2) below. Fig. 3.8 depicts how information can be inferred for the instances using the relations between the corresponding classes.

Additionally, in example (3), the behavior to grasp the wine-glass, that can be done from the cylinder (leg) (according to robot gripper constraint with respect to the diameter of the wine-glass) is described. As shown in Fig. 3.7, the path of grasping should be cleared (i.e. removing the obstacle of Coca-can). There are many scenarios to remove the obstacles, in order to grasp the goal object (wine-glass), such as pick, push or pull. However, with respect to the robot and object constraints, pick and place actions from the body of wine-glass are not possible in this case. In addition, regarding the information in the context class *"Crowd"* (that means the robot will encounter some obstacles), the best scenario is manipulation path (motion), as a sub-task provided by the task planner (i.e. a push action is applied to move the Coca-can to execute the grasp action of the wine-glass).

Example 1: This example describes the query to the knowledge to push a cup.

Question 1: This question describes *"how to push a rigid body?*"

```
?- pushRigidBody( hasAction(action, object) ).
Answer: Pushable object = object.
```

This query requires some information from ontology of "how to interact the rigid body?" as detailed below.

$$
\begin{aligned}
&Cup := rigidbody \\
&\wedge \exists hasSuperclass(PhysicalObject, Rigidbody) \\
&\wedge \exists hasPart(Cup, handle) \\
&\wedge \exists hasPart(Cup, body) \\
&\wedge \exists hasAction(Cup, push) \\
&\wedge \exists hasType(Cup, manipulatable) \\
&\wedge \exists hasInteractionRegion(Cup, body) \\
&\wedge \neg \exists hasInteractionRegion(Cup, handle) \\
&\wedge \exists hasInteractionVelocity(body, velocity) \\
&\wedge \exists hasInteractionParameter(body, physicalProperties) \\
&\wedge \exists hasInteractionParameter(gripper, physicalProperties)
\end{aligned}
$$

Example 2: This example describes the query to the knowledge to pick a cup. In order to apply pick action, another question regarding grasping arises.

Question 2: This question describes *"how to pick a rigid body?*"

```
?- pick( hasAction(action, object) ).
Answer: pickObject = object.
```

$$
\begin{aligned}
&Cup := rigidbody \\
&\wedge \exists hasSuperclass(PhysicalObject, Rigidbody) \\
&\wedge \exists hasPart(Cup, handle) \\
&\wedge \exists hasPart(Cup, body) \\
&\wedge \exists hasAction(Cup, pick) \\
&\wedge \exists hasType(Cup, manipulatable) \\
&\wedge \exists hasInteractionRegion(Cup, handle) \\
&\wedge \neg\exists hasInteractionRegion(Cup, handle) \\
&\wedge \neg\exists hasInteractionRegion(Cup, body) \\
&\wedge \exists hasInteractionVelocity(body, velocity) \\
&\wedge \exists hasInteractionParameter(handle, physicalProperties) \\
&\wedge \exists hasInteractionParameter(gripper, physicalProperties) \\
&\wedge \exists hasGraspingPose(cup, graspingPose)
\end{aligned}
$$

Example 3: This example describes two queries to the knowledge to pick the wine-glass when it is obstructed by an obstacle (Coca-can), set by the task planner.

a. Push obstacle

$$
\begin{aligned}
&Coca-can := rigidbody \\
&\wedge \exists hasSuperclass(PhysicalObject, Rigidbody) \\
&\wedge \exists hasObstacles(Rigidbody, Cocacan) \\
&\wedge \exists hasAction(Cocacan, push) \\
&\wedge \exists hasType(Cocacan, manipulatable) \\
&\wedge \exists hasInteractionRegion(Cocacan, body) \\
&\wedge \neg\exists hasInteractionRegion(Cup, handle) \\
&\wedge \neg\exists hasInteractionRegion(Cup, body) \\
&\wedge \exists hasInteractionVelocity(body, velocity) \\
&\wedge \exists hasInteractionParameter(body, physicalProperties) \\
&\wedge \exists hasInteractionParameter(gripper, physicalProperties)
\end{aligned}
$$

b. Grasp the wine-glass

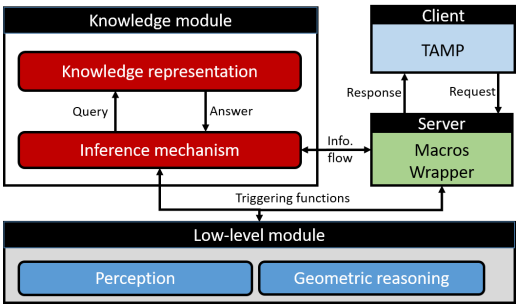


Figure 3.9: Schema showing the integration of the proposed ontology with physics-based motion planning.

$$
\begin{aligned}
&Wine-glass := rigidbody \\
&\wedge \exists hasSuperclass(PhysicalObject, Rigidbody) \\
&\wedge \exists hasObject(Rigidbody, Wineglass) \\
&\wedge \exists hasAction(Wineglass, grasp) \\
&\wedge \exists hasType(Wineglass, manipulatable) \\
&\wedge \exists hasInteractionRegion(Wineglass, cylinder) \\
&\wedge \exists hasInteractionVelocity(cylinder, velocity) \\
&\wedge \exists hasInteractionParameter(cylinder, physicalProperties) \\
&\wedge \exists hasInteractionParameter(gripper, physicalProperties) \\
&\wedge \exists hasGraspingPose(object, graspingPose)
\end{aligned}
$$

Fig. 3.9 shows the base of the implementation of the ontology-based frameworks proposed in this thesis. It is composed of several modules: knowledge-based reasoning, low-level modules, and planning module (e.g., physics-based planning, task planning, ...etc). In the aforementioned examples, the physics-based planning module is used which requires some guidance from the knowledge-based reasoning module by asking some questions. To answer these questions, a reasoning mechanism is required. In order to ground the answers to these questions, Macros wrapper is implemented to facilitate the interface with the physics-based planning module. Moreover, low-level modules such as perception and geometric reasoning to check the action feasibility can be used if required.

3.3.3 Summary

In this work, we presented a knowledge representation in terms of ontologies for physics-based manipulation planning. Instead of making predictions based on inferences, a semantic map is generated to categorize and assign the manipulation constraints using reasoning based on logical axioms. We extended the OUR-K framework with our previous ontology for physics-based manipulation problems, by adding actor class, which describes the robot working constraints. The three examples of $pick$, $push$ the cup and grasp the wine-glass (after removing the obstacle)

are successfully executed using The Kautham Project integrated with ODE dynamic engine. This ontology formulation is being extended with perception features, concepts and procedures.

This work did not cover the required components of task planning like information of how to geometrically execute the task, e.g. reachability and spatial reasoning. Moreover, it did not integrate the perception system to figure out the environment's entities. All these issues are covered in the following section.

3.4 A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation

Perception and Manipulation Knowledge framework (PMK) is a knowledge processing framework that aims at providing knowledge and reasoning for the requirements of TAMP. Moreover, some other semantic information is also provided as a complement to the geometric feasibility of actions (e.g., an action can be geometrically feasible, although it may not be the type of action a human would logically do, like to place a cup on top of a mobile). PMK also aims to include this type of semantic reasoning.

3.4.1 Task and Motion Planning Inference Requirements

Task and motion planning combines the discrete action selection of task planning with the path generation of motion planning. Figure 3.10 abstractly shows some significant necessary requirements that can be provided by the knowledge to the task and motion planning levels. Some questions arise at the task planning level such as: *does a given precondition hold?*, *how should feasibility checks of the actions be performed using semantic attachments? (Dornhege et al., 2009)*, *what is the spatial situation of the objects?* Others arise at the motion planning level such as: *how should objects be manipulated?*, *how should the motion feasibility be checked?*. Many TAMP approaches need to reason about some necessary requirements in different robotics applications. For instance, knowledge-based TAMP has been proposed to solve the navigation among movable obstacles problem (NAMO) of a multi-robot system (Akbari et al., 2018b), where reasoning is done on the feasibility of the push and pull actions needed to move the obstacles away to clear the path towards the goal. A knowledge-based TAMP enhanced with a cloud engine has been proposed in (E. Tosello, 2018), where the knowledge stores robots expertise on manipulation and navigation tasks and the cloud engine provides tools to efficiently retrieve this knowledge to analyze the requirements of new task and motion planning problems. A heuristic-based task and motion planning approach is proposed in (Akbari, 2018) to solve table-top manipulation problems for bi-manual robots, where a relaxed geometric reasoning (regarding for instance reachability issues) is addressed in the computation of the heuristic. A constraint-based approach called Iteratively Deepened Task and Motion Planning, IDTAMP (Dantam et al., 2016b) iteratively generates candidate task plans, checks on their

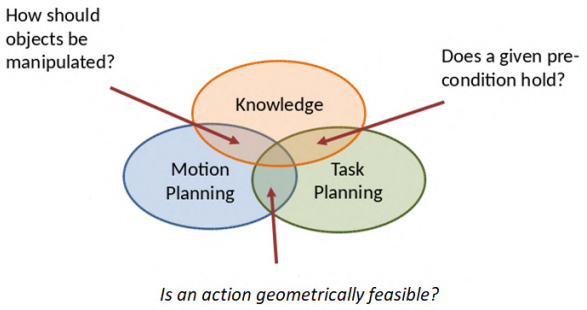


Figure 3.10: Knowledge-based reasoning for task and motion planning (TAMP).

feasibility, and reasons on the failures to further constrain the search.

From the aforementioned approaches, a list of requirements to reason about in TAMP problems can be made: (1) the appropriate interaction parameters (such as friction, slip, or maximum force) required by physics-based motion planners to correctly interact with rigid bodies; (2) the spatial relations (such as *on, inside, right*, and *left*) between the objects on a cluttered scenario; (3) the feasibility of actions due to geometric issues like arm reachability, collisions, or the availability of object placements; (4) the feasibility of actions due to the object features (e.g., overweight objects out of robot capability); (5) the geometric constrains that limit the motion planner space; (6) the action constraints regarding the interaction with the objects (e.g., a *cup* is graspable from the handle and pushable from body); and (7) the initial scene for the planner regarding for instance the potential grasp poses.

PMK framework covers the aforementioned points by providing the reasoning mechanism, based on a perception module associated with the framework, to increase the planning capabilities and fulfill the planning requirements. Moreover, knowledge representation is based on the standard IEEE-1872 to increase the shareability among planning approaches.

3.4.2 System Formulation

A robotic system is proposed, shown in Figure 3.11, which is composed of: a perception module, the PMK framework, a TAMP planning module, and an execution module. In the perception module, the tags are used to detect the poses and IDs of world entities and asserting them to the PMK to build the IOC-Lab [1] knowledge.

A motivation example is included that introduces a manipulation table-top problem at IOC-Lab, as shown in Figure 3.12. Two robots, the dual-arm YuMi robot and the TIAGo mobile manipulator, share the knowledge between each other in order to perform the task of serving

[1] Institut d'Organització i Control de Sistemes Industrials

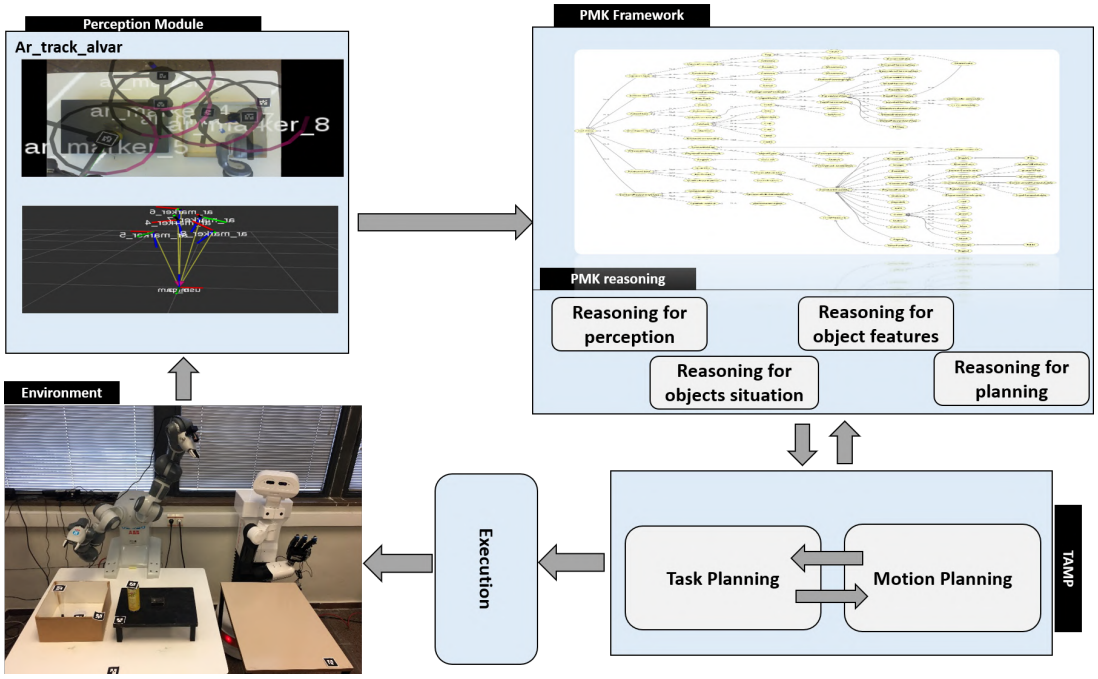


Figure 3.11: Main parts of the system: The perception module, the Perception and Manipulation Knowledge (PMK) framework, and the TAMP planning module. PMK asserts the perceptual data, builds the IOC-Lab knowledge, and provides the reasoning predicates to the planning module.

beverages on a table. The set-up is prepared by YuMi, while TIAGo is used to actually serve the beverages to the customers.

The PMK framework with reasoning mechanisms is used to provide the reasoning predicates related to perception, object features, situation analysis, and geometric reasoning.

These reasoning components, discussed in detail in this section, facilitate the planning process. The TAMP module is a combination of the FF task planner and physics-based motion planning (Akbari et al., 2016b). It is used to actually plan the task and provide a feasible sequence of actions to the robot to be executed. The execution module uses YuMi and TIAGo robots to execute the serving task.

PMK Knowledge Structure

A preliminary version of PMK structure has been presented in (Diab et al., 2017). It was inspired from an ontological schema proposed in (Lim et al., 2011) for navigation tasks in indoor environments, that describes concepts through the use a hierarchy of ontologies composed of three layers: metaontology layer, ontology schema layer, and ontology instance layer as shown in Figure 3.13 and described in Sec. 3.3. Moreover, examples of how information inferred from

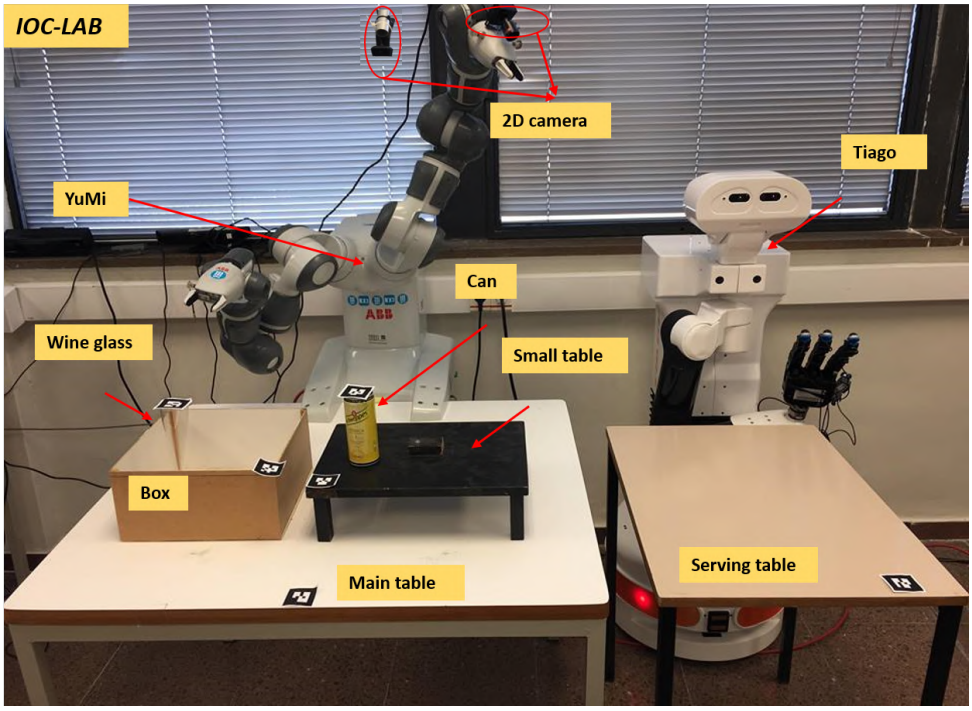


Figure 3.12: A motivation example of a two-robot table-top manipulation task at the IOC-Lab.

these layers are described in Sec. 5.2.

Following this hierarchical schema, we proposed PMK framework [2] for automated manipulation tasks where these layers are composed of seven classes: *Feature, WSobject, Actor, Sensor, Workspace, Context Reasoning,* and *Action* (Some concepts' names have been modified here with respect to how they were presented in (Diab et al., 2017), in order to fit with the current standardized proposal).

A preliminary version of this framework basically provided the concepts, relations, and inference mechanism for physics-based motion planning to teach the robot how to interact with the rigid bodies. However, the information regarding sensing was not considered, concepts that were proposed were not standardized concepts, and no knowledge regarding spatial and geometric reasoning for combined task and motion planning requirements was included. The PMK framework proposed here covers all the above missing issues.

PMK Reasoning Mechanism

The required inference for TAMP consists of: geometric reasoning to determine robot reachability and placement region, manipulation constraints analysis to determine how to

[2] https://github.com/MohammedDiab1/PMK

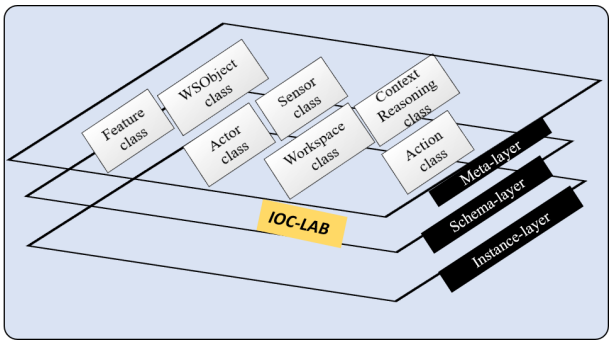


Figure 3.13: PMK ontology.

interact with the objects, and the motion analysis to determine action feasibility.

The reasoning mechanism of PMK is divided into four parts, as shown in Figure 3.11: reasoning for perception, reasoning for object features, reasoning for situation, and reasoning for planning. Reasoning for perception is related to sensors and algorithms to simply answer questions like *which are the sensors the robot has? what is the corresponding algorithm to extract the perceptual data from the sensor?*. Reasoning for object features copes with the features of the objects such as color and dimensions. Reasoning for situation analysis is used to spatially evaluate the objects relations between each other (i.e., cup *inside* box, and *cup* is reachable by *left arm*). Reasoning for planning is used to reason about the preconditions of actions, action constraints, and geometric reasoning (arm reachability, grasping pose reachability, and placement region).

3.4.3 Why PMK?

This section covers the differences between PMK and other knowledge-based processing frameworks, such as KnowRob (Tenorth and Beetz, 2017) and OUR-K (Lim et al., 2011), by highlighting the importance of PMK that is not covered by them. We divide the differences into two levels: modeling and reasoning process. At the modeling level, although these approaches provide frameworks that include a comprehensive way of representing the ontologies related to how to execute manipulation tasks, they lack the representation of the meta-level concepts using the common vocabularies provided by standardization such as (Niles and Pease, 2001b) (i.e., they do not follow any standardization). This may lead to difficulties in incorporating/importing other ontologies under their terminologies because of the conflict in the meaning of the concepts for the robot. This may be a problem for collaborative tasks between robots that require some common vocabularies.

At the reasoning process level, they do not fully cover the area of TAMP to facilitate the planning process. For example, in motion planning, to deal with rigid bodies in cluttered environments, there is the need to define the way to apply actions such as push/pull, requiring a rich semantic description to be fed to the planner, like the physics-based motion planner in (Muhayyudin et al., 2018). In task planning, the robot needs to reason on: (a) the feasibility

of an action at some instant of the manipulation planning process (according to object features, the the state of the object could change and be out of the robot capabilities, e.g., if the cup is empty the state is graspable and if full the state is pushable), (b) the selection of the placement where the robot must place the object, and (c) the current constraints. Moreover, reasoning about perception knowledge is required.

PMK covers these gaps in the modeling and reasoning process levels. In the former, by following the standardized concepts presented by SUMO and CORA. In the latter, by providing reasoning predicates to cover TAMP needs.

3.4.4 Knowledge Formulation

The classes mentioned above in Section 3.4.2 were shown in the metaontology layer in Figure 3.13. The concepts in these classes are formulated here according to the standardized concepts presented in the IEEE 1872 standard. The ontology schema and ontology instance layers are not discussed since they depend on specific domains (in the motivation example, these have been built for the domain of our robotic lab). The formulation according to the standard uses mainly SUMO, CORA, CORAX, and ROA (see Sec. 2.4).

SUMO divides the entities into two groups: *physical* and *abstract*. The *physical* group describes the entities that exist in space-time, and is subdivided into *object* and *process* to represent, respectively, bodies and procedures. The *abstract* group describes the entities that do not exist in time and include mathematical constructs (IEEE-SA, 2015). Following the same modeling strategy, as shown in Figure 3.14, PMK divides the knowledge into knowledge related to objects (manipulation world), knowledge related to processes (manipulation planning), and abstract knowledge (manipulation data). The manipulation world knowledge includes the description of objects, robots and sensors in the workspace. The manipulation planning includes the reasoning processes over PMK. These are detailed in the next subsections. The manipulation data knowledge represents the features of objects, such as color, mass, robot constraints (e.g., joint limits) and sensor constraints (e.g., maximum and minimum measurement ranges).

Manipulation Data Knowledge

It includes the *Feature* class, which is subdivided into three gradual levels to represent data from low-level to high-level (abstract). The standardized concept of SUMO: *Quantity* is used in level one to describe the quantities of the environment, such as object dimensions, color, position, or orientation. A new concept *Quantity Aggregation* has been introduced in level two to include aggregations of quantities, like wrench (force and torque) or pose (position and orientation). The standardized concept of SUMO: *Attribute* is used in level three to represent abstract data like matrices. These concepts are modeled and related to the ontology schema layer (which in our case describes the IOC-Lab domain) to retrieve the required information. For instance, as shown in Figure 3.15, the *Quantity* concept is related to the concept *Pose* to retrieve the information of the grasping pose of YuMi gripper.

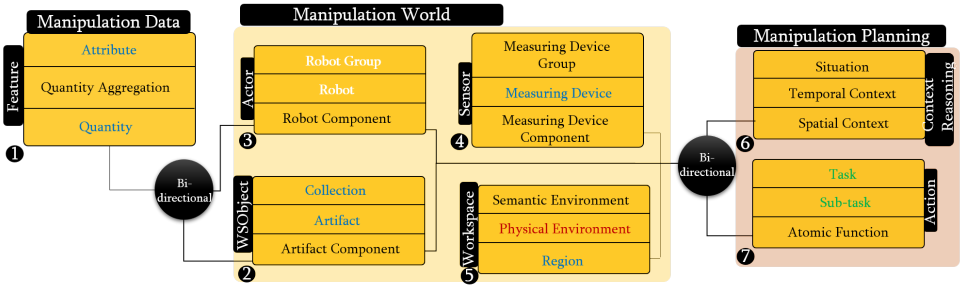


Figure 3.14: Structure of the metaontology layer of PMK fit to follow the standard IEEE-1872. Concepts introduced in this work are shown in black, while those that inherit from the standard are shown in color: SUMO (**blue**), CORA (**white**), CORAX (**red**), and ROA (**green**). All the terminologies are defined in the Appendix.

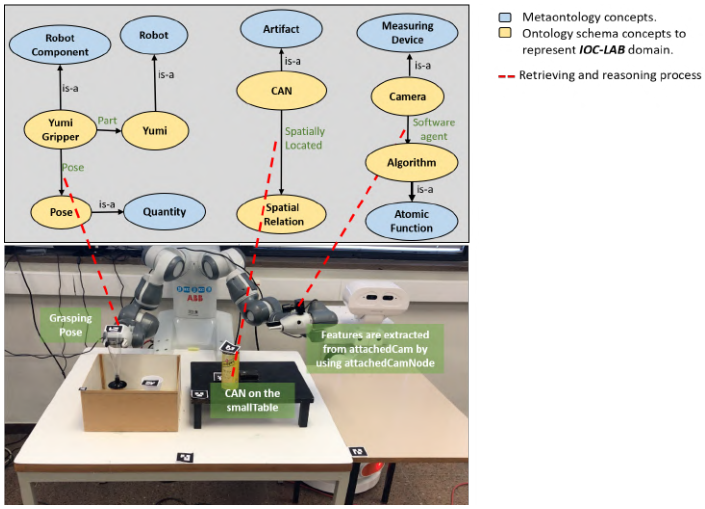


Figure 3.15: Description of some abstract manipulation knowledge concepts.

Manipulation World Knowledge

It includes four classes *WSobject, Actor, Sensor*, and *Workspace*. The main level of the first three classes is level two. In these levels, the standardized concepts are used to define the physical objects and their functionalities, while level one (component-level) represents the description of the components of the objects and level-three (grouping-level) represents the description related to the grouping of the objects in the world. For example, a *cup* has a body and a handle as components, and it can be grouped with a *saucer*.

As shown in Figure 3.14, in the main level of *WSobject, Actor*, and *Sensor* classes, the standardized concepts of SUMO: *Artifact*, CORA: *Robot* and SUMO: *Measuring Device* have been used, respectively. Component-levels use the new concepts *Artifact Component, Robot Component* and *Measuring Device Component*. Grouping-level uses the standardized concepts of

SUMO: *Collection* and CORA: *Robot Group* for the *WSobject* and *Actor* classes, respectively, and the new concept of *Measuring Device Group* for the *Sensor* class. These concepts are modeled and related to the IOC-Lab ontology schema layer to retrieve the required information. Figure 3.15 shows an example of the relation between the component-level of the robot and the physical robot, where the YuMi robot and its gripper are defined under the *Robot* and *Robot component* meta-concepts, respectively.

The main level of *Workspace* class uses the concept from CORAX: *Physical Environment* that describes the topology of the objects in the environment (i.e., which area is occupied by which physical object), the standardized concept SUMO: *Region* is used to describe the geometrical representation of the workspace in level-one, and the new concept of *Semantic Environment* is introduced in level-three to complete level-two with the data (features) of the physical objects.

Manipulation Planning Knowledge

The manipulation planning knowledge represents the part that is responsible for reasoning about the situation of the objects and the robot, and for planning the tasks. This reasoning process is done over the manipulation environment knowledge to facilitate the planning process of the tasks. As shown in Figure 3.14, this knowledge includes two PMK classes, *Context reasoning* and *Action*. Each class has three levels, from high-level to low-level concepts. Level-three represents the symbolic information that depends on the first two levels. For example, in the *Context Reasoning* class, the standardized concept SUMO: *Situation* represents the status of the object or robot in the world with respect to space and time. To cover both, *Spatial Context* (such as *left, right, on, in*) and *Temporal Context* (such as *before, after, meet, overlap*) are introduced to describe them, respectively. These concepts are modeled and related to the IOC-Lab ontology schema layer to retrieve the required spatial information about the environment entities. For instance, Figure 3.15 shows the *can* object, which is sub-class of *Artifact*, that has the property *Spatially Located* to report the ontology with its spatial location with respect to other objects, e.g., the can is *on* the small table.

For *Action* class, symbolic tasks such as *serve* are defined in level-three (*Task*), which are composed of short-term sequences of simpler actions such as *move, pick up, moveholding, place*, that are defined in level-two (*Sub-Task*). The standardized concepts of ROA:*task* and ROA:*subtasks* are used to describe these two levels. Level-one includes the new concept of *atomic function* to represent processes for motion, manipulation, and perception, such as task planners, motion planners, or perceptual algorithms. Moreover, it includes primitive actions, preconditions, and postconditions related to manipulation behaviors. Although ROA provides the concept of *robot behavior* (Olszewska et al., 2017), it does not fully cover the manipulation planning perspective that we need. On the one hand, for sensors, the corresponding suitable algorithms (depending the type of sensor) should be available to extract the features of the environment (see Section 3.4.4). On the other hand, for planning (e.g., task and motion), the corresponding algorithms should be available to plan according to the problem. So, the new concept of *Atomic Function* is introduced to define the algorithms related to planning in terms

of motion and task, and the perceptual algorithms.

Knowledge Representation for Perception

To perceive a robot environment, different sensors are usually used. Sensors provide data about the environment in the form of signals (one dimension) or images (multi-dimension), and to obtain the useful features from the perceived data the suitable algorithms have to be applied, for instance to detect an object pose some pose estimation algorithms based on image features can be applied, or alternatively algorithms based on tags identification can be used.

In Figure 3.14, sensors are defined in sensor class as measuring devices (level-two). The sensors, which can be attached with the robot or fixed in the environment, may contain different parts, which are represented as sensor components (level-one), like the tags, antenna, and reader that contain a RFID sensor. Several sensors may also be grouped as a device group (level-three) for a given application or domain, like the grouping of a RFID sensor and a 2D camera for object localization. This grouping, for instance, allows the robot to understand that it has two types of sensors to locate objects, as well as their differences, e.g., type of data extracted from each sensor, algorithms to be used on the data provided, or the best environmental conditions for their use. The sensor grouping concept makes the robot aware of its available equivalent sensing strategies and allow the selection of the proper sensor to use in each case according to the situation. PMK provides the relation between the *Feature, Sensor*, and *Action* classes that allows to extract the information from sensors. For instance, as shown in Figure 3.15, the concepts of *Camera* and *Algorithm*, which are inherited form *Measuring Device* and *Atomic Function* meta-concepts, respectively, are modeled and related to the IOC-Lab ontology schema layer to retrieve the required algorithm to extract the image features.

The inference mechanism required for the sensing procedure is detailed in Section 3.4.5. The main advantage of this way of representation is that it is workable for any type of sensor and any type of data processing algorithm (like pose estimation from tag detection in 2D images implemented in the ar–track–alvar library (http://wiki.ros.org/ar_track_alvar) and used in the case study shown below).

3.4.5 Case Study

In this section, serving task is proposed to serve a beverage to a customer. The YuMi robot (bi-manual robot) is used to prepare the beverages. It overcomes some manipulation constraints. To tackle the task, a set of different reasoning predicates regarding task and motion planning, perception, are proposed. The description of the task, its constraints, and the implementation tools are described below. Moreover, the reasoning mechanism to facilitate the planning process is introduced. The use of those predicates in the proposed task is explained in detail. The evaluation of the case study and the system flexibility are finally discussed.

Task Description

Consider a manipulation problem including the bi-manual YuMi robot and a set of objects as depicted in Figure 3.12. The task is to *serve* the *wineglass* on the *servingTable*. Initially the *wineglass* is located inside the *box*. Due to reachability limitations, both arms have to collaborate with each other to solve the task. The responsibility of YuMi right arm is to pick up the *wineglass* and place it on the *smallTable* and then, using the left arm, pick up the *wineglass* up and place it on the *servingTable*. The challenge is that the placement region of the *wineglass* on the *smallTable* is already occupied by the *can*. Any planner to be used to solve this task needs a rich semantic description of the scene, able to answer questions such as *what are the sensors the robot has?*, *what are these sensors detecting?*, *what is the associated algorithm to extract the object features?*, *what is the spatial situation of the objects?*, *what is the available regions to place the object?*, or *how can the obstacle be removed?* and some other questions highlighted in Figure 3.16. PMK can provide answers to these kind of questions.

The perception module consists of two 2D cameras and tags for all the objects (see Figure 3.12). One camera is fixed on the top of the main table to sense the main table entities, and the other is attached to the YuMi left arm to perceive the serving table. The tags are used to identify the world entities and semantically link them to the the properties of each object. Specifically, the purpose of the perception module is to detect the position of the objects and their IDs and assert them on the ontology to build the ontology schema and the ontology instance layers of the IOC-Lab environment (as shown in Figure 3.11).

An expressive inference process helps to identify the hidden knowledge and increase the robots capabilities. The PMK framework uses an inference mechanism that consists of Prolog predicates. These predicates are used to query over the ontology to obtain the knowledge that the robot requires to manipulate the objects in the environment. The inference mechanism for manipulation planning domain includes the reasoning process related to sensing, task planning, and motion planning. The related generic predicates are explained in the following subsections.

Implementation

PMK can be integrated with any task and motion planner such as (Akbari et al., 2016a) to compute the sequence of actions to solve a manipulation task. The planner may ask the ontology questions about *how to perform the actions, what are the objects' poses*, or *which are the interaction parameters of the objects?* The PMK handles the requests of the planner and answers by retrieving information, updating/deleting or reasoning over it. The request-answer relation is done using the service-client communication of ROS (Robot Operating System, www.ros.org). The PMK is designed using ontology web language (OWL) with Protégé ontology editor (http://protege.stanford.edu/). Ontology instances can be asserted using information processed from low-level sensory data. The C++ library *ar_track_alvar* (http://wiki.ros.org/ar_track_alvar) has been used to detect the object pose and ID. These data are asserted in the PMK to extract a semantic description of the object. The 2D cameras have been used as a measuring device

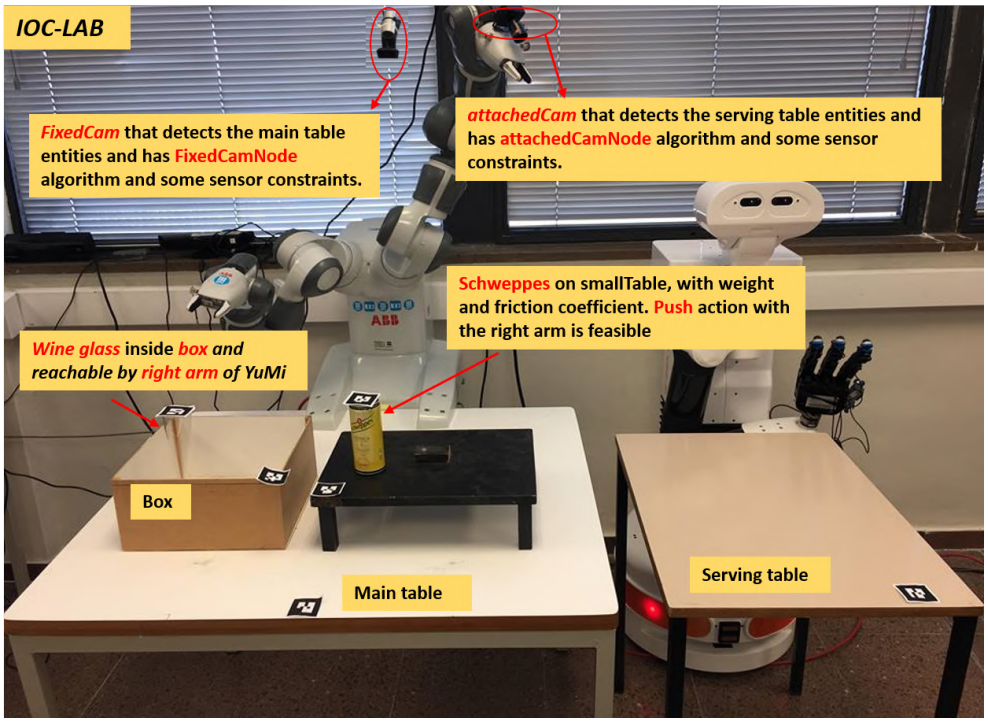


Figure 3.16: The description of the constrains of the motivation example related to sensors, object geometry, interaction learning, and action feasibility.

with two ROS nodes called *FixedCam* and *AttachedCam*, for the fixed and attached cameras, respectively. All the transformations of the objects and camera are calculated with respect to the world frame located at the YuMi base. Queries over the PMK are based on SWI-Prolog and its Semantic Web library which serves for loading and accessing ontologies represented in the OWL using Prolog predicates.

PMK Ontology Representation

In the metaontology layer, *region* is a concept linked to *artifact* and *quantity*, as shown in Figure 3.17 where the relation between the ontological layers that describe the IOC-Lab under the metaontology concepts is illustrated. In the ontology schema layer, the IOC-Lab region has artifacts such as *box, table, wineglass, can, cup* and quantities such as *color* and *dimension*. In the ontology instance layer, the instance *box01* of the subclass *box* represents the storage area that contains the instance *wineglass01* and *cup01* of the subclasses *wineglass* and *cup*, respectively. These instances have perceptual properties such as pose and tagID, that are asserted with the following Prolog predicate:

$$rdf\_assert(instance, registerName\colon objectproperties, assertedvalue),$$

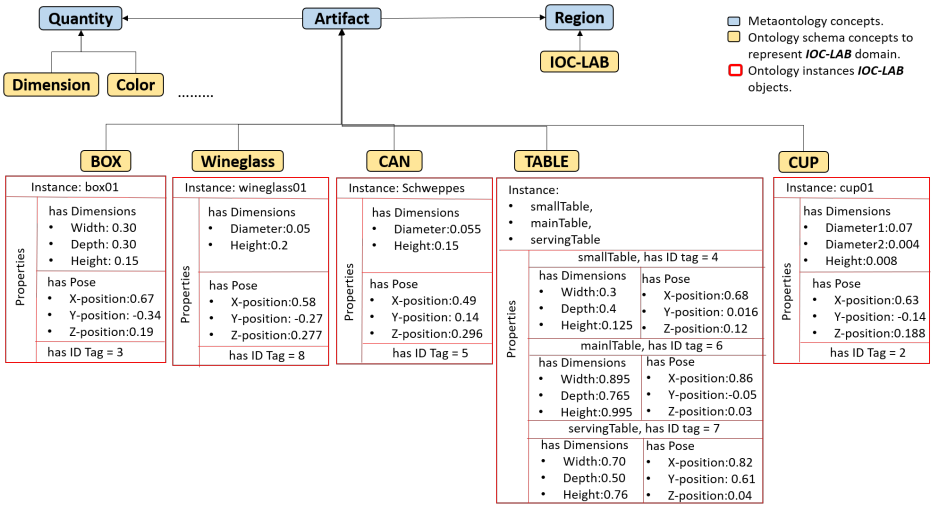


Figure 3.17: Taxonomy representation of the IOC-Lab domain in the PMK framework. The concepts of region, artifact, and quantity are inherited from Workspace, Wsobject, and Feature classes, respectively. The Quantity describes the features of each instance as shown in the properties part in the instantiated artifacts (the orientation information of the objects poses is not included in order to simplify the figure).

and linked to the fixed properties stored on the ontology, so as to have a complete knowledge about the objects. Then, once the the robot figures out the object ID, all the features can be extracted. In order to know where is the reasoning done, who is doing the assertion, and how the assertion is done, see Fig. 3.19.

Reasoning Process on Perception

The inference process related to perception knowledge is basically the reasoning about feature extraction algorithms for perception, such as *FixedCam* or *AttachedCam* nodes used to extract the poses and IDs from images. The inference process related to the perception knowledge of PMK basically depends on the relation between three classes (*Feature, Sensor*, and *Action*). As shown in Figure 3.18, the *device* is a concept that has *quantity* and *atomic function*. The *quantity* contains the constraints and perceptual data. The constraints describe the sensor limitations, such as the fixed camera measuring only the *mainTable* with certain minimum and maximum ranges. The perceptual data describes the type of data and its properties, e.g., pose and IDs are the properties that are extracted from an image. These data are divided into two main parts *Feature of Interest* and *Observable Property* (names are inspired from the Semantic Sensor Network Ontolgy, https://www.w3.org/TR/vocab-ssn/). The former describes the type of data that a sensor senses, such as camera senses images. The latter is an observable characteristic (property) of a *Feature of Interest*, such as color is one of the image properties. As an example, the *RunFixedNode* predicate, shown below using DL, is applied to reason about the location and ID of the objects by being associated with the *FixedCam Node* atomic function (algorithm).

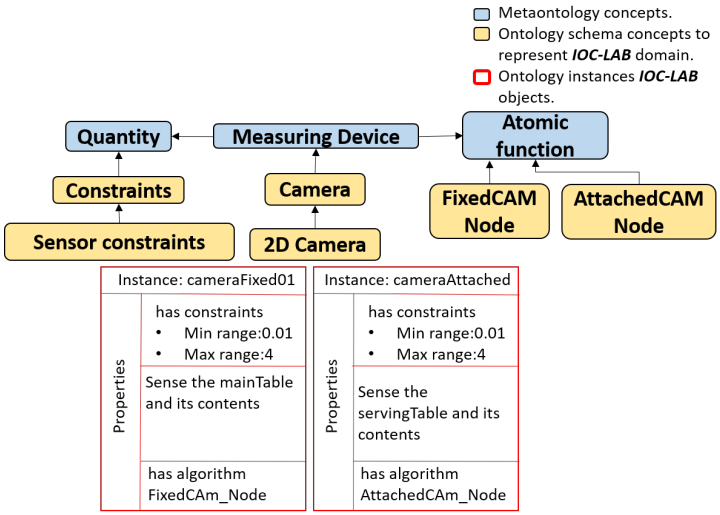


Figure 3.18: Taxonomy representation of sensor class and its relation in PMK framework. The Quantity and atomic Function describe the sensor constraints and the labeled running algorithm of each sensor, as shown in the properties part, and inherit from Feature and Action classes, respectively.

$$
\begin{aligned}
&RunFixedNode :- \\
&\exists hasSuperclass(subTask, Action) \\
&\wedge \exists \textit{Is-a}(FindObject, subTask) \\
&\wedge \exists \textit{Is-a}(FixedCam, measuringDevice) \\
&\wedge \exists hasfeatureOfInterset(2DCamera, Image) \\
&\wedge \exists hasObseredProperty(Image, location) \\
&\wedge \exists hasObseredProperty(Image, ID) \\
&\wedge \exists useProcess(FixedCam, FixedNode)
\end{aligned}
$$

Reasoning Process on Situation

Now, the question that arise is, *what is the situation of the world entities?*. An evaluation-based process inspired from (Tenorth and Beetz, 2009) has been introduced to spatially analyze the situation of the environment (e.g., *box on mainTable, wineglass01 inside box, smallTable left box)*. As shown in Figure 3.19, the spatial relations *right, left, on*, and *inside* have been introduced with some conditions, as detailed in Table 3.1, based on the bounding boxes of the objects in the environment. For example, to make the robot capable to understand that *box01* is *on* the *mainTable*, the predicate must check if the XY-projection of the bounding box of the top artifact lies within that of the bottom one, and then check for the Z-coordinates.

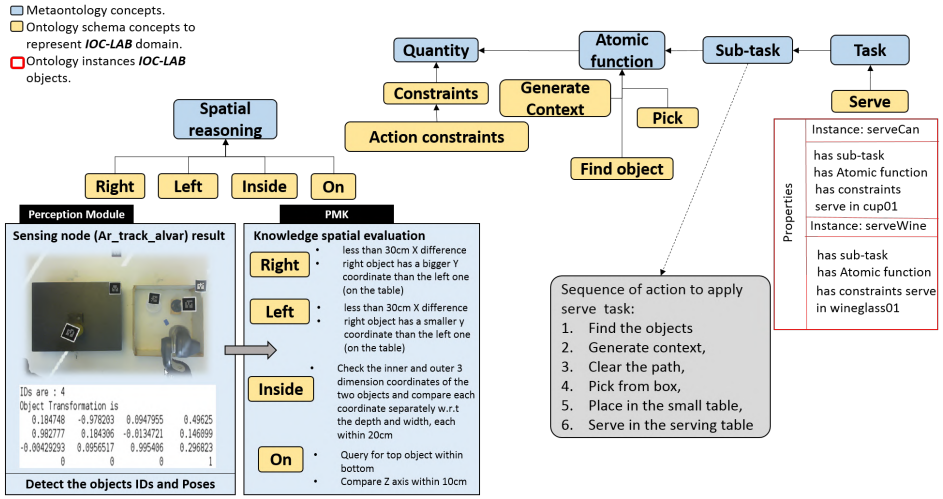


Figure 3.19: Schema of the flow of information between classes that belong to the metaontology, ontology schema, and ontology instance layers of manipulation planning classes. On the left, the flow of asserted data from perceptual module and the spatial evaluation process. On the right, the task and the sequence of actions needed to execute it.

After the situation analysis is done, the PMK can provide useful information regarding the capabilities of the robot, placement regions and action constraints. For example, as shown in Figure 3.12, the tagged placement region of the *wineglass* is the same one that the *can* is occupying. The query to the knowledge to reason about the placement region is:

$$OccupiedPlacementRegion :- \\ \exists hasregionID(obj, ID) \\ \wedge \exists hasspatialRelation(obj, on).$$

Reasoning Process on Discrete Actions

To place the *wineglass*, first the placement region should be cleared from the obstacle *can*. For that, PMK must reason about the robot capabilities with respect to the objects to be manipulated, in order to determine which actions should be involved. The robot has a maximum payload and a given aperture of the gripper, and the objects contain information regarding the parts from where they should be grasped, as well the regions from where they should be pushed (e.g., a region defined around the object and below its center of mass). Then, for instance, to evaluate the feasibility of the pick up action, the size of the part to be grasped with respect to the gripper aperture has to be verified as well as the weight of the object with respect to the robot maximum payload. In this example the YuMi robot is able to pick up the *wineglass* but not the can (due to both its size and weight), which therefore needs to be pushed in order to be removed from the

Table 3.1: Spatial reasoning.

DL Description	Condition	Effect
(On, Subjective, Objective) :- $\exists hasSuperclass(Spatialcontext, Contextreasoning)$, $\wedge\exists hasSubjective(On, artifact01)$, $\wedge\exists hasObjective(On, artifact02)$.	The X and Y ranges of the bounding box of the Subjective artifact must be within those of the Objective, and the Z-ranges must be contiguous.	artifact01 *On* artifact02
(inside, Subjective, Objective) :- $\exists hasSuperclass(Spatialcontext, Contextreasoning)$, $\wedge\exists hasSubjective(inside, artifact01)$, $\wedge\exists hasObjective(inside, artifact02)$.	The X and Y ranges of the bounding box of the Subjective artifact must be within those of the Objective, and the Z-ranges must overlap.	artifact01 *inside* artifact02
(Right, Subjective, Objective) :- $\exists hasSuperclass(Spatialcontext, Contextreasoning)$, $\wedge\exists hasSubjective(right, artifact01)$, $\wedge\exists hasObjective(right, artifact02)$.	The X and Z ranges of the bounding box of the Subjective artifact must overlap those of the Objective, and the X-range of the bounding box of the Subjective must have its minimum value greater than the maximum value of the X-range of the Objective.	artifact01 *right* artifact02
(left, Subjective, Objective) :- $\exists hasSuperclass(Spatialcontext, Contextreasoning)$, $\wedge\exists hasSubjective(left, artifact01)$, $\wedge\exists hasObjective(left, artifact02)$.	The X and Z ranges of the bounding box of the Subjective artifact must overlap those of the Objective, and the X-range of the bounding box of the Subjective must have its maximum value lower than the minimum value of the X-range of the Objective.	artifact01 *left* artifact02

placement region it is occupying. The following DL predicate illustrates the reasoning about the capability of *pick up* action:

$$
\begin{aligned}
&Pickup: - \\
&\exists hasSuperclass(Pick, Action) \\
&\wedge\forall hasTaskTarget(Pick, Artifact) \\
&\wedge\exists hasArm(Robot, Arm) \\
&\wedge\exists hasConstrains(Artifact, Top) \\
&\wedge\exists hasGaspingPose(Artifact, GraspingPose) \\
&\wedge\exists hasObjectPose(Artifact, ObjectPose) \\
&\wedge\exists hasdimension(Artifact, Bbox_1) \\
&\wedge\exists isMemberOf(Gripper, Robot) \\
&\wedge\exists hasdimension(Gripper, Bbox_2) \\
&\wedge\exists fitInside(Bbox_1, Bbox_2) \\
&\wedge\exists hasCapability(Payload, Gripper)
\end{aligned}
$$

Reasoning Process on Motions

As explained in Sec. 3.3, the inference process for physics-based motion planning queries over the knowledge and reasons about the manipulation constraints. These involve the manipulation regions (regions around the object from where the robot is allowed to interact with the objects), and also the interaction dynamics parameters like friction and damping coefficients, and bounce velocities defined for the physics engines. A DL predicate to query to the knowledge to reason about physics-based manipulation constraints is the following:

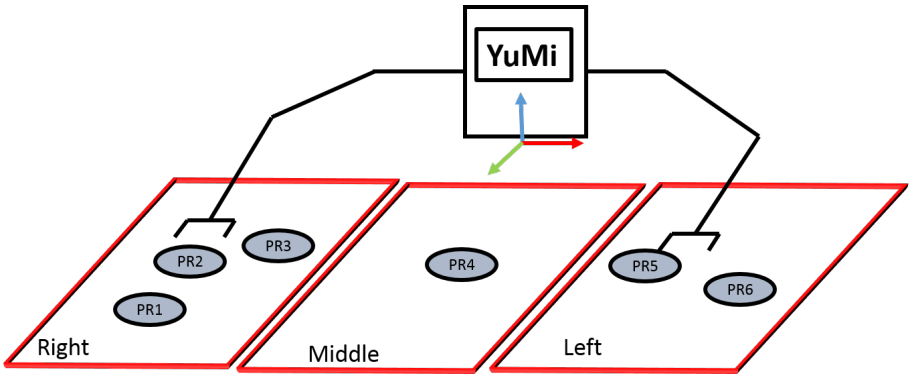


Figure 3.20: The reachability space representation.

$$
\begin{aligned}
&ManipulationConstraints :- \\
&\exists hasSuperclass(artifact, WSobject) \\
&\wedge \exists hasmaterial(material, artifact) \\
&\wedge \exists hasmass(mass, artifact) \\
&\wedge \exists hasInterParameter(interactParameter, artifact) \\
&\wedge \exists hasManReg(manipulatableRegion, artifact)
\end{aligned}
$$

Extended Spatial Reasoning

For grasping and placing the *wineglass*, it is necessary to select the arm to execute the action and the placement region. For example, the right arm is selected for picking up the *wineglass* from the *box*, since it can not be done by the left one due to kinematic limitations. Moreover, to place it in a *smallTable* or *servingTable*, suitable regions are defined. To automatically reason about the reachability, the reachability space is represented as follow.

- *Reachability Space Representation*: As shown in Figure 3.20, the three regions have been introduced; right, left, and middle, and a given set of labels per region indicate where objects can be placed. The right arm is responsible for tackling the objects in the right area, while the left arm is responsible for grappling with the objects in the left area. Both arms can be used to handle the objects in the middle area. The predicate *robot-reachability-grasping(Artifact, ?reachableArm)* has been used to figure out which arm can be used to grasp an object, while *robot-reachability-placement(Arm, ?PlacementTargetRgn)* has been used to figure out the target placement regions to place an object.

As an example, consider a *cup* is placed at *PR1* (Figure 3.20). Then, the predicate *robot-reachability-grasping(Artifact, ?reachableArm)* evaluates to true for the right arm of the YuMi, and false for the left arm. Then the predicate *robot-reachability-placement(Arm,*

?PlacementTargetRgn) can be used to verify that the right arm can place the cup at *PR1*, *PR2*, *PR3*, or *PR4*.

Task Planning and Execution using PMK

This subsection describes the usage of PMK predicates to execute the manipulation task. Figure 3.19 shows the ontology layers regarding the task planning concepts. In the metaontology layer, the relation between *task*, *subTask*, and *atomic function* is shown. In the ontology schema and ontology instance, related to the current example, it is shown that the robot has the ability to *serve* wine or a soft drink by defining two instances *serveWine* and *serveCan* (i.e., pick up the *wineglass* or *cup* from the *box* to start filling them for the customers regarding their demand *serveWine* or *serveCan* respectively). The task is divided into a sequence of Sub-task, while the actual actions and their behaviors are described as atomic functions. In this sense, *Atomic Function* have quantities to define constraints (e.g., which part of the object can be grasped?).

The primary requirement is to run the corresponding algorithm to each camera to estimate the objects position according to the environment reference frame, as shown in Figure 3.16. The predicate *RunFixedNode(FixedNode,? Algorithm)* is running. The result of the query is *FixedCamNode* (as shown in Figure 3.21b) that recognizes images and extracts the pose and ID of the *mainTable* entities. Then, to set the initial scene of the environment to the planner, the poses are used to spatially analyze the situation of each object using the proposed spatial predicates (e.g., *wineglass* inside *box*). The robot task planner queries to ask about which arm can be used to pick up the *wineglass*, and where the *wineglass* can be placed, using *robot-reachability-grasping(Artifact, ?reachableArm)* and *placementRegion (Artifact, ?Region)*. Due to the occupancy of the tagged placement region of the *wineglass* by the obstacle *can*, the task planner queries about the feasibility of available actions to remove the can, using the predicate *feasible(Artifact, ?Action)*. The results of these queries show that the object is *graspable* or *pushable*, according to the payload of the object and the capability of the robot. To know how to interact with the objects, the physics-based motion planning needs the interaction parameters and information from where the robot can interact (i.e., manipulation region). For this, the motion planner uses the *manipulationConstrains(Artifact, ?Properties)* predicate. The sequence of executed actions using these aforementioned predicates is shown in Figure 3.21c–f. Then, as shown in Figure 3.21g,h, the robot picks up the *wineglass* with the right arm and places it at the *servingTable*, according to the results of the predicates *robot-reachability-grasping(Artifact, ?reachableArm)* and *robot-reachability-placement(Arm, ?PlacementTargetRgn)*.

Figure 3.21 has shown the sequence of actions that have been executed in a real experiment, once the planning has been efficiently done with the help of PMK.

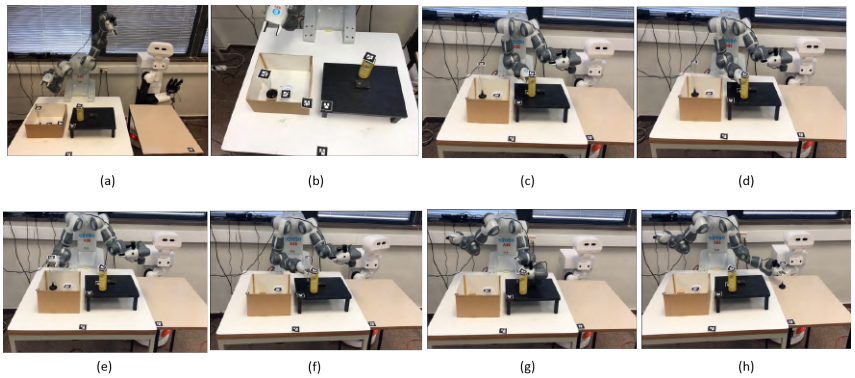


Figure 3.21: The sequence of snapshots for the execution of the *serve* a glass of wine instruction. Video URL-link "https://sir.upc.edu/projects/kautham/videos/PMK-final.mp4".

3.4.6 System Flexibility

To illustrate the capacity to adapt to similar environments that PMK gives to the robot, let us consider the case where the task is to serve a cup of coffee (instead of the wineglass) also located inside the box, and that the region on the small table is currently occupied by a wineglass (instead of by a can), i.e., different objects and at different locations are considered for the same *serve* task (Figure 3.22). The PMK reasoning capabilities allow the robot to handle the differences: (a) now the only feasible grasping configuration for the *cup* is from the top, instead of the lateral grasping configuration used for the wineglass; (b) to remove the wineglass from the intermediate location, a pick up and place action has to be done instead of the push action formerly done for the can.

Like done in the first example, first, the *RunFixedNode(FixedNode,? Algorithm)* predicate is called to run the corresponding algorithm to extract the features and evaluate the spatial relations between the entities, then *robot-reachability-grasping(Artifact, ?reachableArm)* and *placementRegion(Artifact, ?Region)* are applied to know which arm can be used and where the *cup* can be placed. The predicate *feasible(Artifact, ?Action)* can infer that the *wineglass* (which occludes the placement region of the *cup*) is within the robot capability to apply the pick up and place actions to remove it, as shown in Figure 3.22b–d. Then, to know the feasible grasp for the cup, *manipulationConstraints(Artifact, ?Properties)* is used. The sequence of executed actions using the aforementioned predicates are shown in Figure 3.22e–g. Finally, the robot can apply the *pick up* action using the *robot-reachability-grasping(Artifact, ?reachableArm)* predicate to know which arm is reachable, then *robot-reachability-placement(Arm, ?PlacementTargetRgn)* predicate to know where is the placement region to place the *cup* in the *servingTable*.

Therefore, provided that the objects to be manipulated are from the domain included in the ontology schema, the robot is able to adapt to new situations and accomplish the task by means of the perception and reasoning capabilities given by PMK.

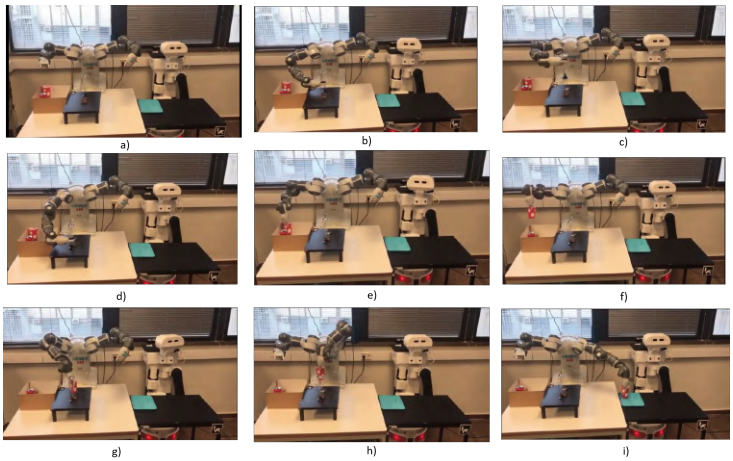


Figure 3.22: The sequence of snapshots for the execution of the *serve* a cup of coffee command. Video URL-link "https://sir.upc.edu/projects/kautham/videos/PMK-final.mp4".

3.4.7 Discussion

Discussion about the Results

The motivation example highlights the significance of using PMK to make the robot understand the current scenario and act according to the actual manipulation constraints. The manipulation constraints related to task execution can be listed as:

1. Manipulation constraints such as: From where the object can be interacted?, What are the interaction parameters?
2. Geometric constraints such as: What is the spatial robot reachability? Where can the objects be placed?
3. Action constraints such as: From where can the actions be applied?
4. Perception reasoning such as: What is the sensor attached to the robot? How does it work? What are its constraints, such as the sensor range measurement?.

PMK provides the answers needed by combined task-motion planners and by physics-based motion planners.

Some other frameworks proposed a knowledge-based processing and reasoning such as (Tenorth and Beetz, 2009; Lim et al., 2011). Although there are some similarities with them in terms of spatial reasoning or reasoning about the object features which could be common in the working domains, the reasoning process related to planning (including geometric reasoning) and perception reasoning are newly proposed in PMK. For instance, these frameworks proposed

some spatial relations such as *on, inside, right, left* for the robot to spatially related between the environment entities. The extended spatial reasoning that includes some extra predicate such as spatial robot reachability and arm selection for bi-manual robots have been introduced. In the case study, the importance of these predicates appear, for instance, when the robot needs to execute the pick up action for *wineglass*. First, by asking about which arm is reachable, then by asking about the reachable placement region to place the *wineglass* on the *servingTable* with the left arm. Moreover, when using a physics-based motion planning, to apply the *push* action, a query is posed to know how to interact with the obstacle *can*, which needs, for instance, the interaction parameters like the friction coefficient.

Moreover, for perception, PMK proposes the reasoning related to the perceptual features of the objects in the environment, like others, but also some reasoning related to the suitable algorithms that the sensor can run to extract the features, the sensors that are associated with the robot, and the sensor features and its limitations. The perception reasoning process makes the robot smarter and more flexible. This flexibility can be useful to cope with the failure of a sensor, by providing an alternative one, i.e., PMK has the flexibility to deal with multi-model sensory systems.

Discussion about the System

The execution of manipulation tasks with knowledge-based planning approaches not explicitly prepared for TAMP, like (Tenorth and Beetz, 2009), can be a challenge because this would require, on the one hand, from the knowledge perspective, to provide all the components in a way that they match with their planning system. On the other hand, from the planning perspective, they would require the definition of the recipes (strategies) for executing the tasks (sequence of actions), including all possible strategies of execution and the way to switch between them when required, which can be a very expensive process, especially for tasks that need long sequences of actions, such as those involving manipulation in cluttered environments. Some other planning approaches rely on PDDL and on planning strategies best fit to cope with difficult task planning challenges, like those found in the manipulation domains, although the use of PDDL implies a closed-world assumption, which precludes their use in more dynamic environments that could require perception and knowledge-based geometric reasoning.

The main role of PMK is to facilitate the planning process for TAMP by providing the necessary planning components, such as geometric reasoning, dynamic interactions, manipulation constraints, and action constraints. Moreover, it includes the perception knowledge and reasoning about the sensors that the robot has, the corresponding algorithms to extract the features, and the sensors limitations. All these components are required to automatically execute complex tasks, like those presented in (Lagriffoul et al., 2018; Quispe et al., 2018), and to make the system flexible enough to adapt to different situations requiring different manipulation actions, as demonstrated with the two motivation examples.

3.4.8 Summary

This study proposed the formalization and implementation of the standardized ontology framework PMK to extend the capabilities of autonomous robots related to manipulation tasks that require task and motion planning (TAMP). In terms of modeling, the aim has been to contribute with a unified framework based on standardization that provides common vocabularies in order to have the flexibility to incorporate PMK with other ontologies, to avoid conflict in the meanings of concepts. Moreover, in terms of reasoning process, some important components for task, motion, or combined task-motion planning are proposed, such as reasoning for perception, reasoning about the object features, reasoning about the environment, geometric reasoning, and reasoning for planning.

To illustrate the proposal, two examples had been introduced to show the PMK framework abilities to query knowledge about the geometric reasoning and robot capabilities to solve TAMP problems. Most of the requirements for TAMP, discussed in this work, are met by the proposed framework. The knowledge is organized in a way to facilitate the planning process, so that the robot can easily access the concepts it needs for its tasks, i.e., PMK enables the robot to complete a manipulation task autonomously in spite of hidden or partial data. To substantiate the perceived information, knowledge related to perception has been considered, which allows analyzing the situation of the environment. Since TAMP queries the requirements to PMK, it automatically adapts to different scenarios.

3.4.9 Enhancement

The PMK framework dose not cope with the navigation problems which requires the extension with more perception measuring devices, such as RFID sensors and 3D cameras, to be applied to more complex manipulation tasks involving assemblies. Also, an extension is planned to include concepts and reasoning related to failures, in order to make PMK useful in situations where the task planner may not be able to find a feasible sequence of actions to perform a given task, or may need to recover from an error. This enhancement has been done in Chapter five.

3.5 Comparison of PMK against other ontology-based approaches to robot autonomy

A comparison between PMK and some projects that use ontologies to support robot autonomy is proposed. The systematic search for projects that fulfill a set of inclusion criteria has been done, and the comparison them with each other with respect to the scope of their ontology, what types of cognitive capabilities are supported by the use of ontologies, and which is their application domain.

3.5.1 Inclusion criteria

Some frameworks or projects have been selected which are considered to be object of the analysis performed in this work. However, among them, the focus is only on the discussion and the comparison on the most influential approaches. Hence, this work provides a list of inclusion criteria to refine the list of surveyed projects. As presented in the Section 4.4 in (Olivares-Alarcos et al., 2019), authors briefly introduce the excluded approaches and provide some justification for our decision. Projects or frameworks are only considered in the scope of this work if they satisfy all of the following criteria:

1. Ontology scope: The project uses an ontology that defines one of the terms shown in Table 3.2, such as Capability, Skill, Plan and Function, etc. that are identified as particularly relevant for autonomous robotics (those terms are presented in Sec. 3.1 in (Olivares-Alarcos et al., 2019));
2. Reasoning scope: It uses ontologies to support robots manifesting at least one of the cognitive capabilities (discussed in Section 3.2 in (Olivares-Alarcos et al., 2019));
3. Transparency: It is transparent. Meaning that some material (e.g., websites, publications) is openly available that describes the overall goal of the project, what cognitive capabilities are considered, and how and what ontologies are used;
4. Curation: It is maintained. Meaning that recent developments or future plans are evident or at least possible; and
5. Accessibility: There exists –at least a prototypical –software that is accessible, and that demonstrates how ontologies are used to support a cognitive capability.

3.5.2 Comparison of abstract concept and domain category

There are six frameworks/projects that satisfy the criteria from different robotics applications areas such as service and industerial, etc., and The PMK (Diab et al., 2019) is one of them. Beside PMK, others like KnowRob (Tenorth and Beetz, 2009, 2017), ROSETTA (Stenmark et al., 2018), IEEE Standard Ontologies for Robotics and Automation (IEEE-SA, 2015), ORO (Lemaignan et al., 2010), CARESSES (Bruno et al., 2017) are discussed in this review. For each of them, their underlying principles and foundations are discussed, as well as what application domain the system was designed for. Also the description of how the frameworks evolved over time, and what impact they have had so far.

As mentioned in Table 3.2, abstract concepts such as objects, environment map, action, capability, hardware, and software are proposed in PMK and some of the other frameworks. Uniquely, PMK proposed some terms like Function and Interaction which are not proposed by other frameworks. As a result of it, PMK is considered as a quite comprehensive and flexible enough to model several robotics domains through it.

Term	KnowRob 1/2	ROSETTA	ORO	CARESSES	OROSU	PMK
Object	Yes/Yes	Yes	Yes	No	Yes	Yes
Environment map	Yes/Yes	No	No	No	Yes	Yes
Affordance	No/Yes	No	Yes	No	Yes	No
Action	Yes/Yes	No	Yes	Yes	Yes	Yes
Task	No/Yes	Yes	Yes	No	No	Yes
Activity	No/No	No	No	Yes	No	No
Behavior	No/No	No	No	No	No	No
Function	No/No	No	No	No	No	Yes
Plan	No/Yes	No	Yes	No	No	Yes
Method	No/Yes	No	No	No	No	Yes
Capability	Yes/Yes	Yes	No	No	No	Yes
Skill	No/No	Yes	No	No	No	Yes
Hardware	Yes/Yes	Yes	Yes	No	Yes	Yes
Software	Yes/Yes	Yes	Yes	No	Yes	Yes
Interaction	No/No	No	No	No	No	Yes
Communication	Yes/No	No	No	No	No	No

Table 3.2: List of relevant terms for the autonomous robotics domain, and their coverage in the different chosen works. Yes and No state for when the term is or not covered by the ontology of the specific framework. Note that in the cases when the term is needed and taken from the upper ontology used within the framework, and/or when the knowledge is captured using a similar term, it is considered that the term is covered. If the upper ontology contains the term but it is not used, we consider that the term is not included.

As shown in Table 3.3, PMK covers several categories in robotics domain. Decision making and choice category is covered because of the description of a robot system which enhances the execution of plans with the support of the PMK ontology. Based on the beliefs about the workspace (reachability of objects, feasible actions to execute, etc.), the system makes decisions about the distribution of actions among different robotic arms, and also about action's parameters, slightly modifying the original plan. The Perception and situation assessment category is covered because of the integration of a tagged-based vision module within PMK. In this module, the tags are used to detect the poses and IDs of world entities and asserting them to the PMK to build the domain knowledge. Then, a reasoning mechanism is used to provide the reasoning predicates related to perception, object features, geometric reasoning, and situation assessment. Particularly, for situation assessment, an evaluation-based analysis is proposed which generates relations between the agent and the objects in the environment based on the perception outcomes, being, these relations used later to facilitate the planning process. The Problem solving and planning category is covered because of the ability of PMK to serve as tool for any planner to reason about task and motion planning inference requirements, such as robot capabilities, action constraints, action feasibility, and manipulation behaviors. The Reasoning and belief maintenance category is covered because of the ability of PMK to generate semantic maps of the robot's workspace enhancing its belief maintenance. By means of computer vision methods, the robot detects objects and its properties (e.g. poses) and, using the ontology,

Cognitive Capability	KnowRob	ROSETTA	ORO	CARESSES	OROSU	PMK
Recognition and categorization	(Beßler et al., 2019)	–	(Lemaignan et al., 2010)	(Menicatti et al., 2017)	–	–
Decision making and choice	–	–	–	(Bruno et al., 2019)	–	(Diab et al., 2019)
Perception and situation assessment	(Beetz et al., 2015a)	–	(Sisbot et al., 2011)	–	–	(Diab et al., 2019)
Prediction and monitoring	(Beetz et al., 2012)	–	–	–	–	–
Problem solving and planning	(Beßler et al., 2018c)	–	–	–	–	(Diab et al., 2019)
Reasoning and belief maintenance	(Beßler et al., 2018c)	–	(Warnier et al., 2012)	(Bruno et al., 2019)	–	(Diab et al., 2019)
Execution and action	(Tenorth et al., 2014)	(Stenmark et al., 2015)	–	(Sgorbissa et al., 2018)	(Gonçalves and Torres, 2015)	(Diab et al., 2019)
Interaction and communication	(Yazdani et al., 2018)	–	(Lemaignan et al., 2011)	(Bruno et al., 2018)	–	(Diab et al., 2017)
Remembering, reflection and learning	(Beetz et al., 2015b)	(Topp and Malec, 2018)	–	–	–	–

Table 3.3: List of cognitive capabilities for the autonomous robotics domain and their coverage in the different chosen frameworks/ontologies. It is possible to find the reference to the articles in which the different reasoning capabilities are addressed using the ontologies.

it stores a symbolic representation of the workspace. A reasoning process over those symbolic beliefs allows, for instance, to make assumptions about abstract spatial relations (e.g. cup on the table). The Execution and Action category is covered because of the ability of PMK to link the action representation in two forms. The former is the representation of the preconditions and effects of each action in an ontological form, then the planner can query over the PMK to retrieve the required information. The latter is to link the PDDL representation form with ontology. For each action, the planner can query over the PMK to retrieve the required information from a PDDL file. The Interaction and Communication category is covered because of the ability of physics-based motion planner to query over the PMK to reason about the interaction parameters with the physical objects in a certain environment.

3.5.3 Enhancement

This study concludes that PMK gets an impact on modeling and reasoning levels. However, it needs to be tested in more planning approaches like knowledge-enabled approaches such as (Beßler et al., 2018c). Moreover, although PMK is an open-source library, it needs to be well-documented.

3.6 Summary of the chapter

In this chapter, a ontological framework called PMK is proposed to provide useful knowledge to guide and facilitate the planning process within a classical-based manipulation planning framework. This planning framework facilitates the combination of task and motion planning (TAMP) approaches which includes Fast Forward (FF), a classical symbolic planning approach to compute the sequence of actions to be done in a certain task, and physics-based motion planning which deals with motions and possible interactions with the objects. The tools proposed to provide useful knowledge to the planning process called Perception and Manipulation Knowledge (PMK) is proposed. It provides, on the one hand, a standardized formalization

under several foundations such as the Suggested Upper Merged Ontology (SUMO) ,and the Core Ontology for Robotics and Automation (CORA) in order to facilitate the shareability and reusability when the interaction between humans and/or robots is done. On the other hand, the inference mechanism is proposed to reason about TAMP inference requirements, such as robot capabilities, action constraints, action feasibility, and manipulation behaviors. Moreover, PMK allows to break the closed world assumption of classical-based manipulation planning approaches. This proposal has been tested for serving task in a table-top manipulation problem.

Chapter 4

An Ontology-based Approach for Failure Interpretation and Recovery in Planning and Execution

This chapter introduces knowledge-based failure interpretation and recovery tools for table-top manipulation problems, especially in the assembly domain. This tool provides a representation way under SUMO and DUL foundations. Moreover, a reasoning mechanism to reason on geometric components is proposed such as the feasibility of actions and motion constraints in a logic-based planning system. This reasoning mechanism requires the integration of low-level geometric planning modules (that include motion planning, collision check, inverse kinematics, and object placement) to feed back the planner on whether the preconditions of actions are met. Therefore, in this chapter, a geometric and failure interpretation and recovery ontologies are proposed, linked with a low-level geometric module, to provide a heterogeneous way of reasoning that helps any planning system. Some motivating examples have been introduced to illustrate the proposal, which has been tested with a logic-based planner.

4.1 Problem statement and proposed solution

Autonomous mobile robot manipulators may not show a robust performance when placed in environments that are not tightly controlled. An important cause of this is that failure handling often consists of scripted responses to foreseen complications, which leaves the robot vulnerable to new situations and ill-equipped to reason about failure and recovery strategies. Instead of libraries of hard-coded reactions that are expensive to develop and maintain, more sophisticated reasoning mechanisms are needed to handle failure. This requires an ontological characterization of what failure is, what concepts are useful to formulate causal explanations of

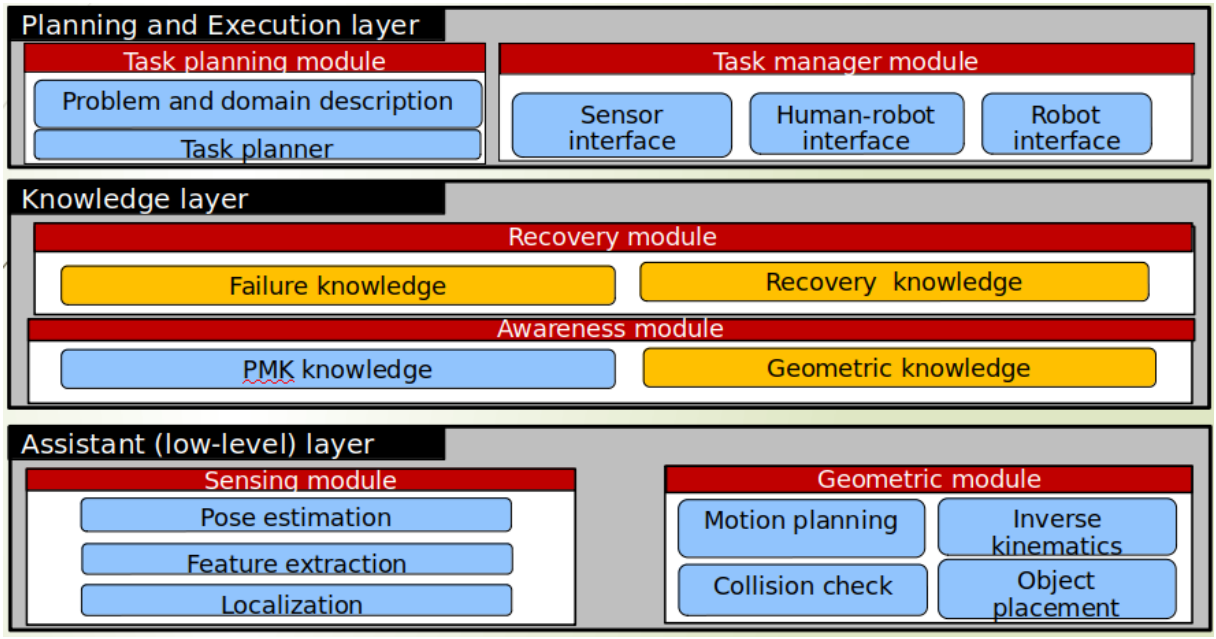


Figure 4.1: FailRecOnt- The combination of the planning system for assembly tasks with the proposed geometric reasoning and failure interpretation/recovery capabilities in orange.

failure, and integration with knowledge of available resources including the capabilities of the robot as well as those of other potential cooperative agents in the environment, e.g. a human user. We propose the FailRecOnt framework as a step in this direction. We have integrated an ontology for failure interpretation and recovery with: 1) a logic-based planning framework in assembly domain, 2) a heuristic-based task and motion planning framework, such that a robot can deal with uncertainty, recover from failures, and deal with human-robot interactions.

4.2 FailRecOnt Framework

In this section, an overview of the framework, the description of the proposed modules in the low-level and the knowledge-level, and how the data is managed are explained in detail.

4.2.1 Overview

The proposed framework – FailRecOnt – is the continuation of KTAMP framework presented in chapter 3, and it is composed of the same three main layers (planning and execution layer, knowledge layer and assistant layer) although now the knowledge layer has been extended, as shown in Fig. 4.1.

The knowledge layer contains a set of knowledge to guide the planning and execution layer:

1. Awareness module: that contains a) PMK knowledge as described in detail in chapter 3; b) Geometric knowledge to provide the geometric reasoning responsible for checking the feasibility of the actions.
2. Recovery module, that provides knowledge to interpret the failures and proposes recovery strategies like: 1) asking a human for assistance for unsolvable tasks by the robot, 2) guiding the robot to autonomously recover itself, for instance by calling a sensing module to figure out the current scene of the world or 3) keep repeating the same action with another parameter (e.g., repeat a grasping action with a different angle).

The next subsections are organized to facilitate capturing the contributions of this chapter. First, Sec. 4.2.2 describes the integration of the proposed geometric ontology to give access to the geometric reasoning module. The main advantage of this is that, instead of calling the module manually from the task planning module, as done in the previous chapter, the robot can query over the knowledge to retrieve the sequence of processes required to execute such actions in an automatic way. However, this requires the implementation of a mechanism that has the ability to figure out the failures that may occur. For this purpose, failure interpretation and recovery ontologies are proposed in Sec. 4.2.3 where the failure concepts, modeling and causes are described.

4.2.2 Geometric knowledge

Concepts describing geometric queries

An ontological module is defined to cover geometric notions used in robotics such as collision checking, placement feasibility, etc. This ontology also describes concepts like geometric querying, query status, and status diagnosis, and contains terms for particular queries such as IK (inverse kinematics) for a specific robot. Some important terms in the ontology are briefly detailed [1]:

- **Geometric querying** is defined as an *Event* in which some spatial reasoner– e.g. a collision checker– participates, and which is classified by/executes a *GeometricReasoningTask*.
- **GeometricReasoningTask** is defined as a classification of the Geometric querying. It is represented as:

  $GeometricQuerying \sqsubseteq (\exists isClassifiedBy.GeometricReasoningTask) \sqcap (\exists hasParticipant.SpatialReasoner) \sqcap (=1 hasStatus.QueryStatus)$.

  We also say that $GeometricReasoningTask \sqsubseteq \forall\ isExecutedBy.GeometricQuerying$.

[1] (Description logic is used, see Appendix B for details.)

- **Spatial reasoner** is defined as a software component used to answer geometric queries. For example, for checking reachability, an IK solver can be used as a *SpatialReasoner*. *SpatialReasoner* is represented as a *SpatialReasoner*$\sqsubseteq$*ComputationalAgent*. Several types of *SpatialReasoner: CollisionChecker, IKSolver, MotionPlanner* are proposed in this ontology.
- **Query status** is defined as the outcome of a *GeometricQuerying*. If failure, it means the query was not answered at all, and the reason is given. Such reasons include the geometric component responsible for the query not answering in time or being unavailable. If success, it means the query has been answered, and the answer can further be interpreted to ascertain what it implies for an action's feasibility. Note that "query failure" is used for situations where no answer at all is given; "geometric failure" for situations where an answer is given, but it is not satisfactory for some constraint. E.g., an IK solver returning it failed to find a solution is not a query failure (the IK solver works well) but it is a geometric failure (a *ReachabilityFailure*).
- **Status diagnosis** It is the information to support the analysis of geometric query answers.

How to call the geometric reasoning module from the geometric ontology

A set of Prolog predicates are implemented to access the geometric reasoning module. Some examples of the Prolog predicates for the geometric reasoning which call the low-level modules (assistant system in FailRecOnt framework) are:

"Is the path toward a goal configurations reachable?":

```
?- testReachability( hasAlgorithm(IKModule, IKAlgorithm) ).
Algorithm = Call IK service.
```

"Is the path toward a goal configurations collision-free?":

```
?- CollisionCheck( hasAlgorithm(CCModule, CCAlgorithm) ).
Algorithm = Call collision check service.
```

Also, these can be a part of a general workflow to call IK, collision check and motion planning in a sequence as follow:

```
?- testReachability( hasAlgorithm(IKModule, IKAlgorithm) ),
?- CollisionCheck( hasAlgorithm(CCModule, CCAlgorithm) ),
?- MotionPlanning( hasAlgorithm(MPModule, MPAlgorithm) ).

Algorithm = Call [IK, CC, MP].
```

However, to consider the possibility of failures occurring while calling the predicates (e.g., the query not getting a response at all), or the outcome of the query showing infeasibility (e.g, the goal configuration being not reachable), a failure interpretation and recovery module is required to be integrated with geometric ontology. This integration aims to facilitate interpreting such failures, their causes, and provide recovery strategies (maybe one or more). The final idea

is to integrate these ontologies with the planning system in order to be able to repair the plan instead of replan when a failure occurs.

4.2.3 How to describe failures: basic concepts of the failure ontology

The basic higher-level concepts in the proposed recovery module are presented here. The module consists of two ontologies: failure ontology and recovery and include the geometric knowledge as a sub-module. The combination is required when reporting the geometric failures in the manipulation domain. The ontological concepts introduced below are absolute terms that cope with the modeling formalisms. Moreover, some basic concepts, without being formally defined here, like *Agent, Plan, Task, Goal* or *Event* are used (as defined in (Niles and Pease, 2001a; Masolo et al., 2003)). The two ontologies concepts are described below using description logic (DL).

The concepts introduced here are formally defined via their relations to foundational ontologies (DUL and SUMO).

a) Task failure: The evaluation of the result of the robot actions in a certain situation based on its expectations. This situation is described by a *Failure narrative*, see below.

b) Failure Narrative: A communicable description of a *Task failure* situation. It defines several roles, to be filled by an *Agent*, *Task*, and a *Goal*. It uses a *Failure symptom*, see below, to classify the *Action* that the *Agent* performed, and may provide an *Explanation* for the failure in a *Failure diagnostic*, see below.

c) Failure symptom: A simple classification label which can be applied to events in order to interpret them as a failure of some sort. Failure symptoms include software error codes and signal or exception types. For example, when a robot controller raises a "hardware error signal", we would say the robot classified an event as being a hardware failure. Error status codes in query returns are another subtype of failure symptom; e.g., the status code that an IK solver would use to indicate it found no solution. Sometimes software signals or exceptions come with more data attached to them in order to identify the failure cause; for example the hardware error signal might also include which motor is thought defective.

Also, a taxonomy of failure symptoms is defined, as follows. One dimension of classifying failures is the "when" of happening:

- **Inception failures:** prevent an action from starting; e.g. *CapabilityFailure*,
- **Performance failures:** prevent an action from completion; e.g. *ResourceDepletionFailure*,
- **End state failures:** the outcome of a completed action is not conformant to the goal; e.g. *ConfigurationNotReached*.

Failures are also classified by the nature of participants. Currently there are three top-level classes for this dimension: **Cognition failure**, **Communication failure**, **Physical failure**.

- **CognitionFailures:** classify events involving only an *Agent* and whatever internal representations it uses,
- **CommunicationFailures:** classify an *Event* that involves some *Agents* exchanging information, and
- **PhysicalFailure:** classify events with only *Physical object* or *Agent* participants; e.g., the robot not being able to manipulate an object because of it being too far (*ReachabilityFailure*) or because the gripper is broken (*EndEffectorFailure*).

For failures where it makes sense to identify a physical location, there is a classification along the "where" of occurrence. So far, this is only seen in the taxonomy for embodied *Physical agents*, and failures may be classified as **Body part failure** (example, *TorsoFailure*).

Finally, there is a dimension of classifying failures along the "what" of the relevant interaction. So far, this is done only for *PhysicalFailures*, which can be **Mechanical failures** or **Electrical failures**.

d) Failure diagnostic: The process is triggered by a detected deviation from the expected behavior of a given system, whose aim is to identify the origin of the failure, i.e. to locate all contributors for the unexpected event. Failure diagnosis includes geometric failure diagnosis regarding reachability, placement region and motion planning. For example, there are many reasons of reachability failures which means the robot can not reach a final pose to grasp/place an object, due to some causes such as:

1. no IK solution exists;
2. IK solution exists but collision with fixed obstacle(s) occurs;
3. IK solution exists but collision with movable obstacle(s) occurs.

For placement region, there are several reasons why failure occurs:

1. no feasible placement (i.e., placement region is too small for the object size);
2. feasible placement exists but collision with fixed obstacle(s) occurs; These causes are listed under this concept to facilitate the classification process of the failures.
3. feasible placement exists but collision with movable obstacle(s) occurs.

For motion planning, a failure means no collision-free path could be found. The causes could be:

Table 4.1: Modeling the failure ontology under the DUL and SUMO foundations.

Concept	DUL	DL description	SUMO	DL description
Task Failure	Situation: A view, consistent with ('satisfying') a Description, on a set of entities.	Situation $\sqsubseteq$ $\exists$satisfies.Description TaskFailure $\sqsubseteq$ Situation TaskFailure $\sqsubseteq$ $\exists$satisfies.FailureNarrative	Propositional attitude: An IntentionalRelation in which an agent is aware of a proposition.	TaskFailure $\sqsubseteq$ PropositionalAttitude TaskFailure $\sqsubseteq$ $\exists$satisfies.FailureNarrative
Failure symptom	Event type: A Concept that classifies an Event. An event type describes how an Event should be interpreted, executed, expected, seen, etc., according to the Description that the EventType isDefinedIn (or used in)	EventType $\sqsubseteq$ Concept EventType $\sqsubseteq$ ($\forall$classifies.Event) FailureSymptom $\sqsubseteq$ EventType FailureSymptom $\sqsubseteq$ $\forall$classifies. ($\exists$hasParticipant.Agent)	Class: similar to Sets, but not assumed to be extensional, i.e. distinct classes may have the same members. Membership decided by some condition.	FailureSymptom $\sqsubseteq$ Class FailureSymptom $\sqsubseteq$ $\forall$ instance^{-1}.AgentPatientProcess
Failure Narrative	Narrative: A descriptive context of situations	Narrative$\sqsubseteq$Description FailureNarrative$\sqsubseteq$ Narrative	Proposition: An abstract entities that express a complete thought or a set of such thoughts.	FailureNarrative $\sqsubseteq$ Proposition
Failure diagnosis	Diagnosis: A Description of the Situation of a system, usually applied in order to control a normal behavior, or to explain a notable behavior (e.g. a functional breakdown).	Diagnosis $\sqsubseteq$ Description FailureDiagnosis $\sqsubseteq$ Diagnosis	Proposition: An abstract entities that express a complete thought or a set of such thoughts.	FailureDiagnosis $\sqsubseteq$ Proposition

1. planner timeout due to collision with movable obstacle(s) (by removing them collision-free path can be found);

2. planner timeout due to collision with fixed obstacle(s).

Modeling of failure ontology under different upper level foundations

Aiming at making the failure ontology sharable and widely used, it has been formalized under SUMO (Niles and Pease, 2001a) and DUL (Masolo et al., 2003) foundations, as shown in Table 4.1. The upper-level ontologies SUMO and DUL foundations are used because of their abilities to cover several concepts related to robotics but also more general domains. Several ontology frameworks are defined under those upper-levels, such as (Diab et al., 2019), to facilitate the incorporation/importing of different ontologies with the same common vocabularies while avoiding semantic conflicts. This way of modeling becomes essential in large-scale research and development projects.

4.2.4 How to identify failures: situation modeling and analysis

Ontologically speaking, the labels "success" and "failure" are perspectival. They do not apply to actions or events per se but are used by an agent to evaluate the result of its action based on its expectations. The robot acts rationally in the world with the aim to bring about a desired state of the world, therefore it needs to establish whether the action it has just performed and the resulting state are as expected. The lack of correspondence between the expected result and the actual state of the world is labeled a failure.

However, this is not yet precise. What a robot can compare is what it knows about the actual vs. what are characteristics of the expected state. In both cases we talk of state descriptions. The robot is not interested in every aspect of the world, but must verify whether the world state satisfies constraints relevant to the robot's goal(s).

This implies the world knowledge used to decide on action failure vs. success is robot-dependent; robots do not all have access to the same ways of describing the world, because of, e.g., different sensors. Hence, an ontological treatment of failure must include robot-specific knowledge. However, a common core of general knowledge about failure exists and can be formalized at a level that abstracts away from robot particulars.

In this work, we concentrate on the case studies on an actual robot platform, and we therefore showcase robot-specific knowledge and more abstract knowledge about failure interacting to inform failure recovery decisions. This should illustrate how the approach is applicable to a broad class of cases, but without implying that it would work for any robot or scenario without some adaptation to the specifics of a robot platform.

In order to assess the success/failure of an action, the following pieces of information must be available for consideration by the robot:

- the last executed action,
- the state of the world before the last action,
- the expected state after the action,
- the actual state after the action and
- possibly knowledge about what might have prevented reaching the expected state.

We describe situation a fragment of physical reality, a state of the world, to the extent that it can be described by the robot's language; a situation is a state of the world modulo what the robot cannot describe. If, e.g., the robot cannot detect and represent colors, a situation is a fragment of reality that omits colors. For a positive example, let us assume the robot's language is as follows: two object identifiers A and B, terms for locations X, Y, etc., and object properties *locatedAt*, *connected*. A situation is a fragment of reality in which A and B are located at some position (say, X and Y, respectively) and may or may not be connected. Suppose they are. The situation is described as:

1. A is located at position X,
2. B is located at position Y,
3. A and B are connected.

Consider two states of the world which are as before and such that in the first it is sunny and in the second raining. In the robot's view these two states of the world are the same situation since the robot's language would not be able to establish a difference between them.

To each situation corresponds a situation description, i.e. an expression in the robot's language giving complete information about the situation. We say that a situation S satisfies a situation description D when S makes the description D true. The situation description of the previous example corresponds to formula (or any equivalent formula):

$locatedAt(A, X), locatedAt(B, Y), connected(A, B)$

A situation can have partial description which provides only some of the information that the robot could have expressed about a situation. E.g., from the previous formula one might have as a partial description:

$locatedAt(B, Y), connected(A, B)$

The partial description of such situations is useful because even if the robot could describe something about the world, it doesn't have to, and often cannot. There may be unknown but knowable facts, such as in the location of an item.

Finally, an expected transformation is a triple $<S_i; Act_i; S_f>$: the execution of action Act_i at state S_i is expected to terminate in state S_f. Transformations may be regularly updated as a result of experience.

Given an action, we say that it is a success if it is an instantiation of the expected transformation $<S_i; Act_i; S_f>$. Instead, a failure is an instantiation of a triple $<S_i; Act_i; S'>$such that (a) the latter is not an expected transformation, and (b) for any expected transformation $<S_i; Act_i; S_f>$, S' and S_f are logically contradictory.

The discussion above shows how we believe one should think about failure detection as a process of matching descriptions in a robot's cognition to fragments of the physical world.

Note, we use here a feature of open-world semantics: facts not asserted are not false, simply unknown. A robot may know of a *Situation* without any *manifestsIn* links to *Event* in the robot's history but, unless that *Situation* is explicitly asserted to be a *NonrealizedSituation*, the robot maintains the possibility that the *Situation* could manifest into an *Event*, e.g. one that hasn't happened yet. Therefore, decisions on whether a *Situation* corresponds to some observed *Event* should come from some other module, such as perception. What the ontology describes here is how the decision from perception should be formalized so that it becomes accessible to further reasoning.

Recovery strategies

A recovery strategy is a method a robot can apply to repair or reconstruct a plan whose execution resulted in a failure. Because replanning is a time-consuming operation, instead, repairing the plan could be an option. Some recovery strategies have been proposed such as:

1. *Repeat last action:* A robot could either employ a “repeat last action” strategy or instead have to address the underlying failure cause. We can axiomatically encode that a ”repeat last action" is not appropriate when the failure has a sustained cause.

2. *Remove failure cause:* Depending on the nature of the underlying cause of a sustained failure, we can use the ontology to formulate new subgoals as part of a *RemoveFailureCause* strategy by checking which of the robot-known situations get classified as *UnrealizedPrecondition*, and hence the unrealized preconditions become goals for new planning queries.

3. *Ignore failure:* A failure is not necessarily a problem for continuing a plan. If the expected outcome is an object located at X, but the object is at Y, this might not be a problem if the object is not needed again and not in the way for other actions. Geometric reasoning beyond the scope of the ontology decides whether an object is “not in the way”. We say an *IgnoreFailure* strategy may be appropriate only when the situation actually manifesting *prevents* no other expected situations. In order to check whether the *prevents* relations hold or not – so as to ascertain whether *impedes* holds or not – we would have to defer to other reasoning modules.

Competency questions

The proposal is to exploit the ontology to discover information helpful for the robot to evaluate and, when needed, overcome a failure. When a failure is detected, the ontology should be able to provide answers to the following failure questions (FQ) we imagine the robot raises:

1. Why was an *Action* classified as failure?

2. Is the failure causing a problem for subsequent activities?

3. If the failure is causing a problem (and so must be addressed), does it matter why it happened and if so what is the causal explanation for the failure?

4. If the failure is causing a problem, what are appropriate recovery strategies?

5. Assuming a recovery strategy is pursued, how can successful recovery be assessed?

Regarding FQ1, this ontology formalizes concepts to organize the robot's information about the action executed and the expected vs. actual situation, and identify aspects of the mismatch e.g. failure location or functional aspects such as involved resources.

The robot's various reasoning modules, such as motion planning and navigation, cooperate in answering FQ2. This cooperation is guided by the ontology in that it formalizes queries about prevention relations between ongoing situations resulting from the robot's past actions and future situations corresponding to goals being achieved, queries to be handled by mechanisms appropriate to the nature of those situations; e.g. navigation can check whether a particular object placement prevents a base movement.

FQ3 reveals how the ontology converts information about prevention relations (in this case, between situations that have already happened and ongoing to situations that are postconditions of the failed action) into selection criteria for recovery strategies. In particular, it allows distinguishing between causes that must be addressed, and hence require defining new subgoals, from causes that can be ignored.

FQ4 is answered by the ontology by the criteria it defines for recovery strategy applicability. These criteria are necessarily incomplete, in that we often can say when a recovery strategy is *not* appropriate, but this in itself is not proof that the recovery strategy actually *is* appropriate. Ultimately, it is the robot's recovery attempt that is the final judge. However, reasoning guided by the ontology can filter out infeasible candidate strategies.

Finally, for FQ5, the recovery strategies defined in the ontology also describe what counts as a success, because "success" is dependent on the strategy and the original transformation expectation. For example, a failed goal is simply abandoned by an *IgnoreFailure*, whereas new subgoals are defined for *RemoveFailureCause*.

4.3 The integration of the proposed ontologies with a logic-based planning

In assembly applications in table-top manipulation problems, assembly recipes can be used and elegantly be represented in description logic theories (see Sec. 2.7 and Appendix B). With such a recipe, the robot can figure out the next assembly step through logical inference. However, before performing an action, the robot needs, on one hand, to ensure various spatial constraints are met, such as that the parts to be put together are reachable, non occluded, etc. On the other hand, if failures occurred while planning or/and during the execution phase, the robot must be able to recognize them, their sources and how to recover them. Such inferences are very complicated to support in logic theories, but specialized algorithms exist that efficiently compute qualitative spatial relations considering the possibility of fail, such as whether an object is reachable.

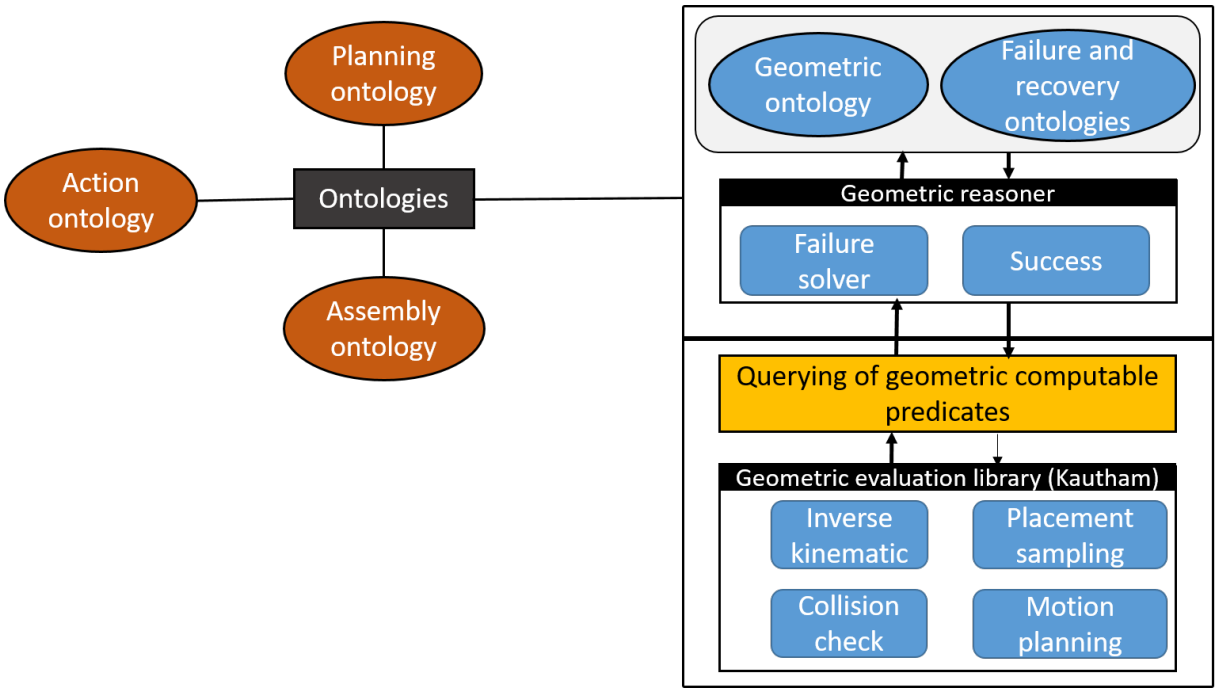


Figure 4.2: Proposed Integration. The brown ontologies provide by KnowRob group (Beßler et al., 2018a), while the blue and yellow boxes are proposed for this integration.

In this chapter, as shown in Fig. 4.1, the FailRecOnt framework is proposed, which combines a logic-based planner, described in Sec. 2.7, with geometric reasoning and failure interpretation/recovery capabilities, to enable robots to perform their tasks under spatial constraints. The geometric reasoner is integrated into the logic-based reasoning through decision procedures attached to symbols in the ontology. Moreover, each assembly step is formally defined in a workflow that structures the step, and that can be automatically executed in a declarative knowledge base.

As shown in Fig. 4.2, the integration of geometric and recovery module is proposed for the geometric failures and the behavior of the robot when encountering the aforementioned failures. This integration is useful to analyze the inconsistencies if they exist, using the feedback of the geometric computable predicates (a reasoning process that reports the failures) that are capable to evaluate those inconsistencies. The Geometric reasoner consists of two submodules; the failure solver and success. The former is used to reason over the Geometric analysis ontology about the the failure types. Moreover, it can provide solution(s) by calling the corresponding module from the ontology, if it exists, or provide different solutions to solve the failure. The latter is used to report the feasibility of actions.

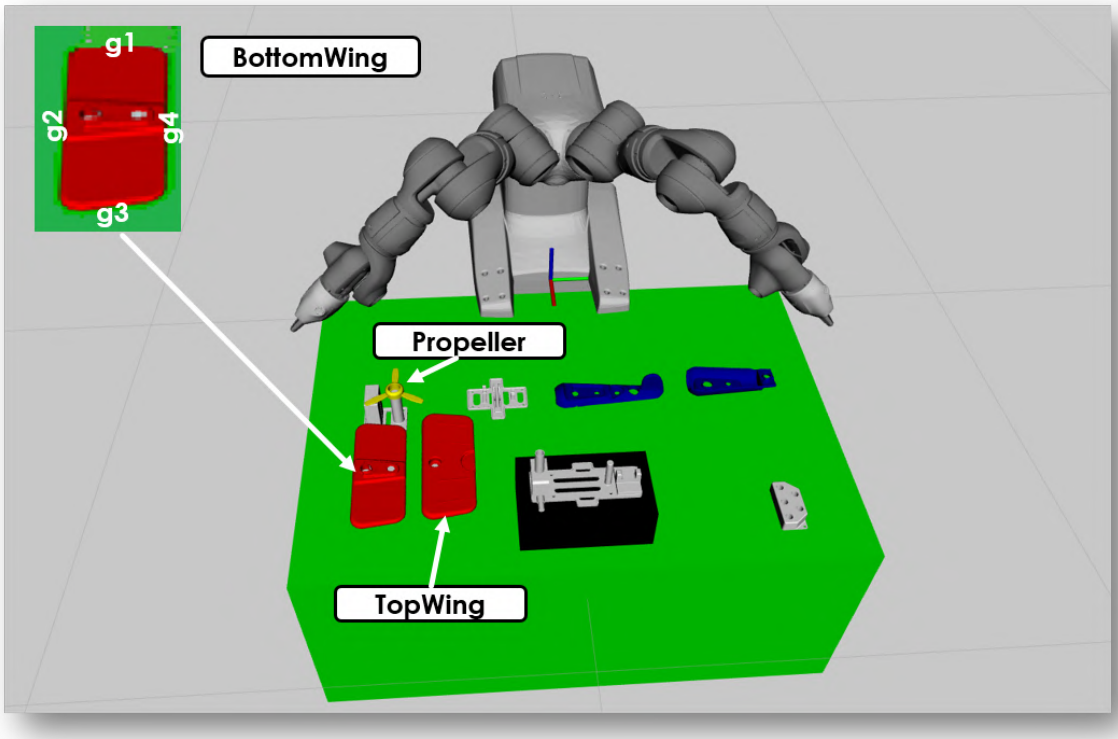


Figure 4.3: Motivating example in assembly domain showing cases that need the planner to use the geometric, failure and recovery ontologies to interpret query and action results. Description of reachability and collision problems for the red objects (top and bottom wings).

Motivation example

We are mainly interested in assembly manipulation tasks for bi-manual robots, which often encounter complexity or failures in the planning and execution phases. Planning phase failures typically refer to failures of the planner itself, but we will use planning phase failures to also refer to situations where the planner reasons that some action would be infeasible, e.g. because objects block access to what the robot should reach. A correct selection of grasps and placements must be produced in such an eventuality. Depending on the type of problem, goal order must be carefully handled especially in the assembly domain; very large search spaces are possible, requiring objects to be moved more than once for achieving the goals. Execution phase failures refer to hardware failures related to the system devices e.g. robot or camera needs to be re-calibrated, or software failures related to the capabilities offered by specific software components, or failures in action performance such as an unexpected occluding object, or slippage.

We selected a toy plane assembly, as shown in Fig. 4.3 targeted at 4-year-old children for the

experimental evaluation. The toy plane is made of 21 plastic parts that are mainly put together using a loose slide in connections, and fixed with bolts afterward. The parts are comparably huge such that grasping them is easier. We use a dual armed YuMi robot and further test the assembly planning in The Kautham Project (Rosell et al., 2014) which is an open-source project for motion planning. It provides the exibility to plan motions under geometric, and physics-based constraints. For the proposed experiments, we use the RRT-Connect motion planner (Kuffner and LaValle, 2000), (Gillani et al., 2016), and the inverse kinematics approach developed by Zaplana (Zaplana et al., 2018). Kautham further provides a ROS-based interface, and the YuMi robot has also a set of existing software services that we use for controlling the robot.

Ontologies are encoded using the Web Ontology Language (OWL) (Antoniou and van Harmelen, 2004), and designed using the Protégé [2] editor.

Assembly Activities in Cluttered Workspaces

Assembly tasks often have a fixed recipe that, if followed correctly, would control an agent such that available parts are transformed into an assembled product. These recipes can elegantly be represented using description logic. But inferring the sequence of assembly actions is not sufficient for robots because actions may not be performable in the current situation. This is, for example, the case when the robot cannot reach an object because it is occluded. A notion of space, on the other hand, is very complicated in a logic formalism, but specialized methods exist that efficiently compute qualitative spatial relations such as whether objects are occluding each other.

The proposed solution is depicted in Fig. 4.2. We build upon an existing planner and extend it with a notion of action, and geometric reasoning capabilities.

Actions are represented in terms of the action ontology which also defines action pre-conditions. Pre-conditions are ensured by running the planner for the action entity. This is used to ensure that the robot can reach an object, or else tries to put away occluding objects. To this end we integrate a geometric reasoner with the knowledge base. The interfaces of the geometric reasoner are hooked into the logic-based reasoning through procedural attachments in the knowledge base. More details will be described in Sec. 4.4.1

4.4 Modifications on knowledge representation of assembly

Some modifications in the ontological relation have been introduced to enhance the logic-based planning presented in section 2.7 in order to smoothly reason over the proposed ontologies.

[2] (http://protege.stanford.edu/)

To connect two parts, they must be in the correct fixture for the intended connection. It could be the case that a fixture blocks a required affordance. In that case, the part should be moved into another fixture that exposes the required affordance, such that the required affordance is exposed.

To ensure this, we assert that required affordances must be unblocked: we add another axiom that restricts the $assemblesConnection$ relation of the action:

$$\forall \textit{assemblesConnection}.(\forall \textit{usesAffordance}.\textit{FreeAffordance}) \tag{4.1}$$

It asserts that any affordance used by the connection must not be blocked. This enforces the robot to use a fixture that exposes the required affordance.

Finally, we define $\textit{partOccludedBy} \equiv \textit{hasAffordance} \circ \textit{occludesAffordance}^{-}$ which relates parts to parts occluding them, and assert that parts cannot be occluded by other parts when the robot intends to put them together:

$$\forall \textit{assemblesPart}.(\leq 0 \textit{partOccludedBy}.\textit{MechanicalPart}) \tag{4.2}$$

This is used to make the robot put away parts that occlude other parts that provide required affordances for this action.

4.4.1 Heterogeneous reasoning process

The proposed reasoning system is heterogeneous, which means that different reasoning resources and representations are fused into a coherent picture that covers different aspects, as shown in Fig. 4.3 and described in detail in Sec. 4.2. In this section, we will describe the two different reasoning methods used: geometric reasoning and knowledge-based reasoning.

a) Geometric Reasoning

The main role of geometric reasoning is to evaluate geometric conditions of symbolic actions. Two main geometric reasoning processes are provided:

Reachability Reasoning: A robot can transit to a pose if it has a valid goal configuration. This is inferred by calling an Inverse Kinematic (IK) module and evaluating whether the IK solution is collision-free. The first found collision-free IK solution is returned, and, if any, the associated pose. Failure may occur if either no IK solution exists or if no collision-free IK solution exists.

Spatial Reasoning: We use this module to find a placement for an object within a given region. For the desired object, a pose is sampled that lies in the surface region, and is checked for

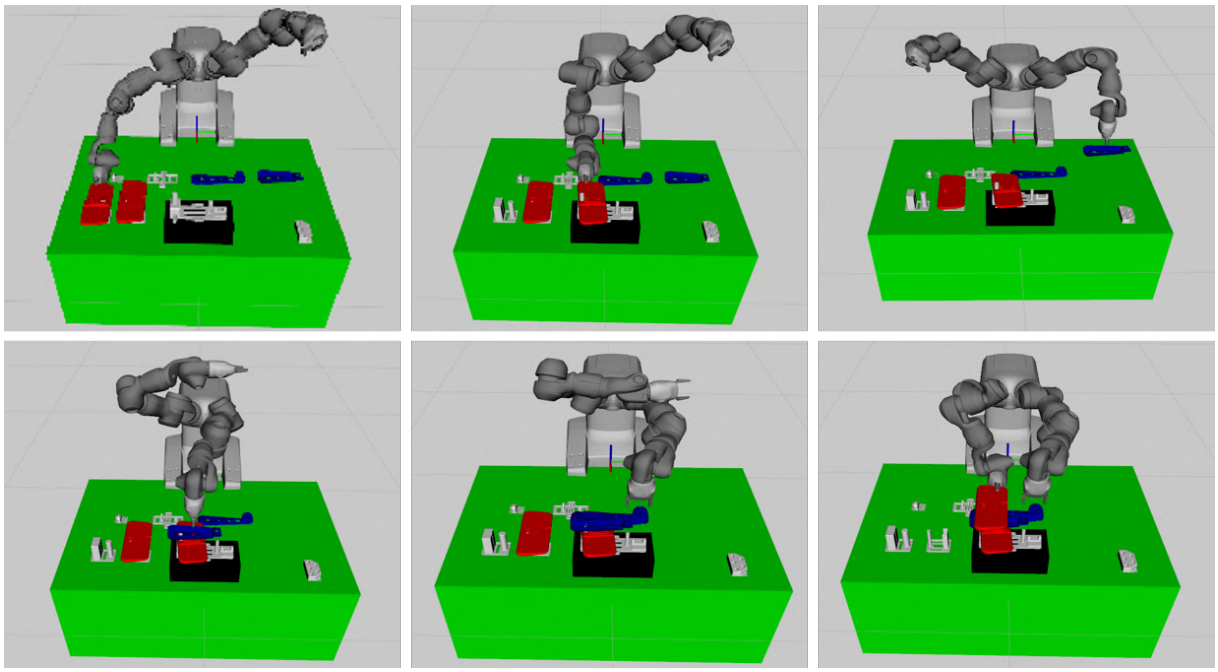


Figure 4.4: The sequence of assemble the Battat toy. The scene has been modeled using PMK ontology.

collisions with other objects, and whether there is enough space to place the object. If the sampled pose is feasible, it is returned. Otherwise, another sample will be tried. If all attempted samples are infeasible, the reasoner reports failure, which can be due to a collision with the objects, or because there is not enough space for the object.

b) Knowledge-based Reasoning

In this project, knowledge-based reasoning refers primarily to checking whether an individual obeys the restrictions imposed on the classes to which it is claimed to belong, identifying an individual based on its relations to others, and identifying a set of individuals linked by certain properties (as done when identifying which parts have been linked, directly or indirectly, via connections).

The geometric reasoner is integrated through computable geometric relations. The robot can then reason about them by asking questions such as *"what are the occluded parts required in a connection?"*:

```
?- holds( needsAffordance(Connection,Affordance) ),
   holds( hasAffordance(Occluded,Affordance) ),
   holds( partOccludedBy(Occluded,OccludingPart) ).
Occluded='PlaneBottomWing1', OccludingPart='PlaneUpperBody1'.
```

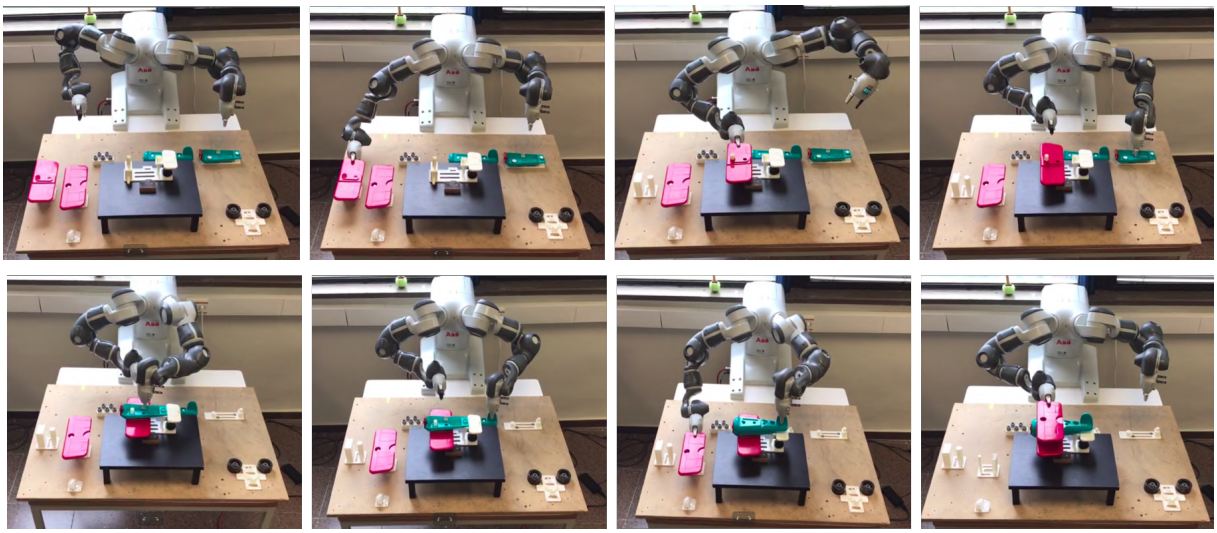


Figure 4.5: YuMi plane assembly execution sequence.

The robot can also reason about what action pre-conditions are not fulfilled, and what it can do to fix this. This is done by creating a planning agenda for the action entity that only considers pre-condition axioms of the action:

```
?- entity(Act, [an,action, [type, 'ConnectingParts'],
      [assemblesConnection, Connection]]),
   agenda_create(Act, Agenda),
   agenda_next_item(Agenda,Item).
Item = "detach PlaneBottomWing1 partOccludedBy PlaneUpperBody1"
```

The robot can reason about what action it should perform that establishes a planning decision in its belief state. It can, for example, ask what action it should perform to dissolve the *partOccludedBy* relation between parts:

```
?- holds( usesActionMapper(Strategy,Mapper) ),
   property_range(Mapper,mapsItem,Pattern),
   individual_of(Item,Pattern),
   call(Mapper,Item,Action).
Action = [an,action, [type, 'PutAwayPart'],
      [movesPart, 'PlaneUpperBody1'], ...].
```

4.4.2 Case study: Use of failure ontology in robotic assembly manipulation planning

To illustrate the proposal the assembly of the Battat toy is proposed, as shown in Fig. 4.3, Fig. 4.4 shows snapshots of the resulting plan in simulation and real environment, respectively. The system is tested with different spatial constraints.

Geometric reasoning about occlusions allows the robot to know when it needs to move parts

out of the way and change the action sequence provided by the logic-based planner.

Some geometric situations are used for testing. The sequence to plan the assembly operations is:

1. call IK module to check reachability for grasping the objects;
2. call a collision checker to validate a path to grasp an object.

Some failures can happen and get reported to the planner, where the failures are interpreted and a decision on the next action is made. Some situations in manipulation domain may happen often, such as the case where an object is blocking the chosen configuration to grap/place an object. This situation requires the selection of alternative feasible (or reachable) grasping poses and/or placements.

For example, as shown in Fig. 4.3, if the object *BottomWing* has four grasping poses from each side g1-g4, the g1 and g4 are occluded by the holders of *PropellerHolder* and *TopWingHolder* respectively, that means g1 and g4 are not feasible and the robot is not able to reach object *BottomWing* through them. Meanwhile, the robot may not be able to grasp the object *BottomWing* through g2 and g3 because of the infeasibility of IK configurations. By querying over the proposed ontology, the robot will be able to analyze the cause and report to the planner.

The failure symptom produced when planning to use g2 and g3 for grasping is a *ReachabilityFailure*, which is a *CapabilityFailure*, whereas a failure produced when planning to grasp using g1 or g4 is an *OcclusionFailure*, which is an *AffordanceFailure*. Both of these are *InceptionFailures* which prevent the planned task from even being undertaken. *CapabilityFailure* can be addressed by generating new capabilities, e.g. selecting new grasping poses that are reachable. Affordance failures, meanwhile, can be addressed by manipulating the environment to better expose its affordances, so by generating intermediary goals of moving the occluders out of the way, the robot can eventually grab the *BottomWing*.

To interpret the causes of those situations presented in Fig. 4.3, the geometric ontology is integrated with the failure ontology as described in Fig. 4.6. This ontology describes the failure symptoms, as well as the *FailureNarratives* which make use of these symptoms to classify failures. The narratives may include other information to enhance the diagnosis process, such as the initial goal and participating objects in the agent's task, and a diagnostic to indicate which component failed. Failure symptoms are ontologically characterized also in terms of what failure diagnostics they are compatible with; for example, a *ReachabilityFailure* can only be used by a failure narrative where the explanation role is played by a failure diagnostic that names some IK component as the failure cause. These modules (i.e, IK and collision check) are low-level modules that the symbolic level considers to be spatial reasoners.

For more explanation on how to use the FailRecOnt framework, another example is proposed

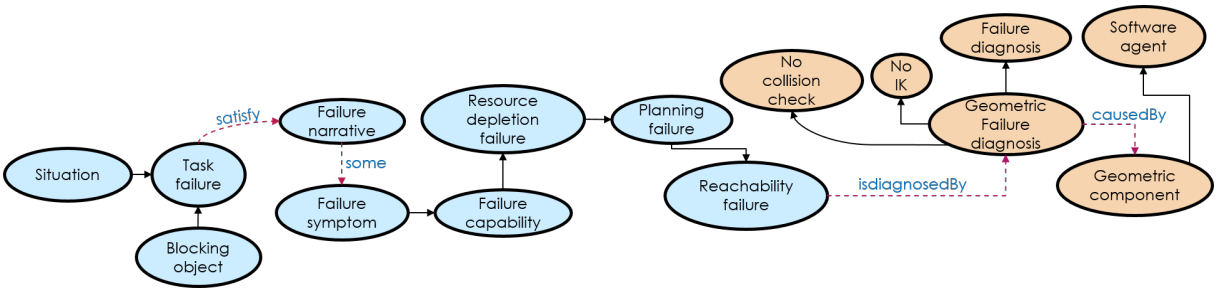


Figure 4.6: An interpretation of blocking object using the proposed failure ontology. The concepts in blue belong to failure ontology, meanwhile the ones in yellow are from the geometric ontology.

(see Sec. 5.9.4 Using recovery module within SkillMaN) in the next chapter using the results achieved in this chapter.

4.5 Summary of the chapter

This chapter proposes:

1. The formalization and implementation of the failure interpretation and recovery ontologies as well as geometric ontology to extend the capabilities of autonomous robots related to manipulation tasks that require task and motion planning along with execution.

2. The integration of geometric ontology with failure interpretation and recovery ones, which is required for the combination of geometric and symbolic levels of planning to interpret such failures.

3. The heterogeneous way of reasoning which improves the robot capability to execute complex manipulation tasks.

   In the modeling level, the absolute concepts of the aforementioned ontologies are modeled under DUL and SUMO foundations to facilitate the usability for roboticists community. A case study is introduced to illustrate the use of failure ontology in automated planning and workflow execution phases by proposing the common situations that could be encountered by such planner.

The aforementioned ontologies have the interface to access the low-level geometric modules to feed them back the required qualitative spatial reasoning.

4.6 Enhancement

This work can be enhanced by developing a strategy to monitor the outcome of the manipulation action in the execution phase. Based on the monitoring process, the robot can detect the failures, if they occur, providing the recovery action.

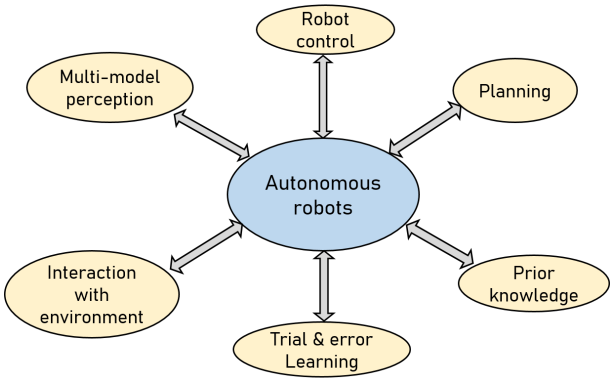
Chapter 5

A Skill-based Robotic Manipulation Framework based on Perception and Reasoning

One of the problems that service robotics deals with is to bring mobile manipulators to work in semi-structured human scenarios, which requires an efficient and flexible way to execute every-day tasks, like serve a cup in a cluttered environment. Usually, for those tasks, the combination of symbolic and geometric levels of planning is necessary, as well as the integration of perception models with knowledge to guide both planning levels, resulting in a sequence of actions or skills which, according to the current knowledge of the world, may be executed. In this chapter, a planning and execution framework, called SkillMaN, for robotic manipulation tasks based on multi-sensory system is proposed, which is equipped with a module with experiential knowledge (learned from its experience or given by the user) on how to execute a set of skills, like pick-up, put-down or open a drawer, using workflows as well as robot trajectories. The framework also contains an execution assistant with geometric tools and reasoning capabilities to manage how to actually execute the sequence of motions to perform a manipulation task (which are forwarded to the executor module), as well as the capacity to store the relevant information to the experiential knowledge for further usage, and the capacity to interpret the actual perceived situation (in case the preconditions of an action do not hold) and to feed back the updated state to the planner to resume from there, allowing the robot to adapt to non-expected situations. To evaluate the viability of the proposed framework, an experiment has been proposed involving different skills performed with various types of objects in different scene contexts.

The significant value of the SkillMaN is the need to make the robot aware of situation similarities to efficiently re-use previous known ways to execute actions by adapting the robot motions to stored patterns (like opening one drawer once the robot knows how to open another). This requires integration of multi-sensory perception, planning in task and geometric

Figure 5.1: Autonomous robots requirements.



levels, reasoning, and learning. SkillMaN exploits the previous works of the perception and manipulation ontologies introduced in chapter three, and the failure and recovery ontologies introduced in chapter four. This integration has been described in detail in Sec. 5.6.

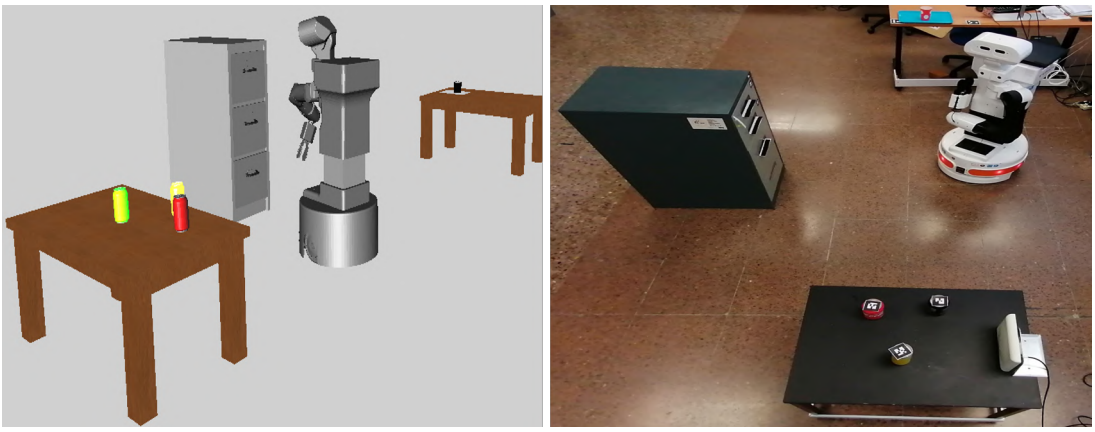
5.1 Problem statement and proposed solution

Mobile manipulators acting as robot co-workers are required to work autonomously in human environments and in the presence of human operators. As illustrated in Fig. 5.1, autonomy can be achieved with the integration of key capabilities, such as planning that usually combines task and motion planning capabilities in order to find feasible plans for the robot to solve complex tasks, and perception responsible for achieving the successful execution of such plans and react accordingly if changes are required. For the former, the use of prior knowledge based on experience become essential to facilitate the planning process in every-day tasks, as well as an adaption process of the robot's skill and trajectories whenever the situation to be faced is similar to a previously encountered one. Moreover, the use of learning-based techniques aiming for autonomous self-improvement can be considered to recover such failures that may occur in the planning or execution phases. For the latter, rich perception modules can be equipped with sensory integration mechanisms to work with different sensors, including for instance those that can cope with non-line-of-sight situations, like RFID.

With this in mind, this study proposes a framework to make the robot work efficiently in semi/unstructured environments in every-day tasks by providing the capabilities for:

1. perceiving objects that may be in the line of sight or not,
2. analyzing the current situation to know if a similar one was previously encountered,
3. planning with the aid of symbolic and geometric reasoning procedures, and

Figure 5.2: The motivation example. On the left, the represented scene in Kautham Project. On the right, the real scene.



4. acquiring experiential knowledge on how to execute a set of skills, to be used for adapting the motions of the robot in similar situations.

All these services and modules are used through a task manager as described later in a way that can be smoothly used in different tasks.

5.1.1 Motion examples

The mobile manipulator TIAGo is assumed to work in the semi-structured environment shown in Fig. 5.2, where there are several *cans* (some filled and some empty) and a file cabinet with three drawers (the first to store empty cans and the second to store the filled ones). Two tasks are scheduled, one task is to sort the cans and store them on the corresponding drawers, and another one is to take a stored filled can and serve its contents to a customer. A perception system is used to figure out the location and status of the objects. To execute these tasks, some skills are introduced: *pickUp*, *putDown*, *openDrawer* and *serving*. The execution of these skills will require, in new situations encountered for the first time, the call to a motion planner to find the robot motions, and in situations similar to previous ones, the use of motion adaptation, i.e. perception, situation similarity checks, planning and the use of experiential knowledge will be required.

To autonomously execute the above-mentioned tasks, a task manager is used to integrate the proposed modules of perception, planning and reasoning in a way that the robot can be adapted working in different environments with minor changes. This means the robot needs to have a description of the environment and then it can use the task manager for coordinating the modules to execute such indoor manipulation tasks.

5.2 SkillMaN framework overview

In this section, an overview of the framework, the description of the proposed modules in the low-level and the knowledge-level, and how the data is managed are explained in detail.

With the aim of executing tasks like that of the motivation example automatically, the integration of several layers and modules is required, covering perception, knowledge representation and reasoning, and planning at symbolic and geometric levels.

The proposed framework – SkillMaN – is composed of three main layers, as shown in Fig. 5.3: planning and execution, knowledge, and assistant (low-level) layer.

- The planning and execution layer contains two modules, the task planning and the task manager modules as discussed in detail in chapter 3.
- The knowledge layer contains a set of knowledge to guide the planning and execution layer.

    1. Awareness module, that contains a) PMK knowledge as described in detail in chapter 3; b) Geometric knowledge to provide the geometric reasoning responsible for checking the feasibility of the skills, as described in detail in chapter 3; c) Skill knowledge to check the availability of the skills in the knowledge database to be used by the planning module, and how to execute them.
    2. Experiential knowledge module, that contains knowledge for planning and situational knowledge. The knowledge for planning provides the geometric-skills information (based on the robot's experience) such as *how to grasp an object? What are the constraints of a task?* For example, if an object is stored in a box or a drawer, the robot requires to reason over the knowledge to figure out which type of grasp is feasible to be successfully executed (i.e., side or top-grasp). Besides this knowledge for planning, situational knowledge is required to check the similarity between the current situations with those stored in a database.
    3. Recovery module, that provides knowledge to interpret the failures and proposes recovery strategies like: 1) asking a human for assistance for unsolvable tasks by the robot, 2) guiding the robot to autonomously recover itself, for instance by calling a sensing module to figure out the current scene of the world or keep repeating the same action with another parameter (e.g., repeat a grasping action with different angle) as discussed in chapter 4.

- Finally, the assistant layer provides the low-level modules that allow to deal with:

    1. perception issues, like finding out which sensors can be used for a sensing action in a given situation, which are dealt by the sensing module, as discussed in chapter 3.

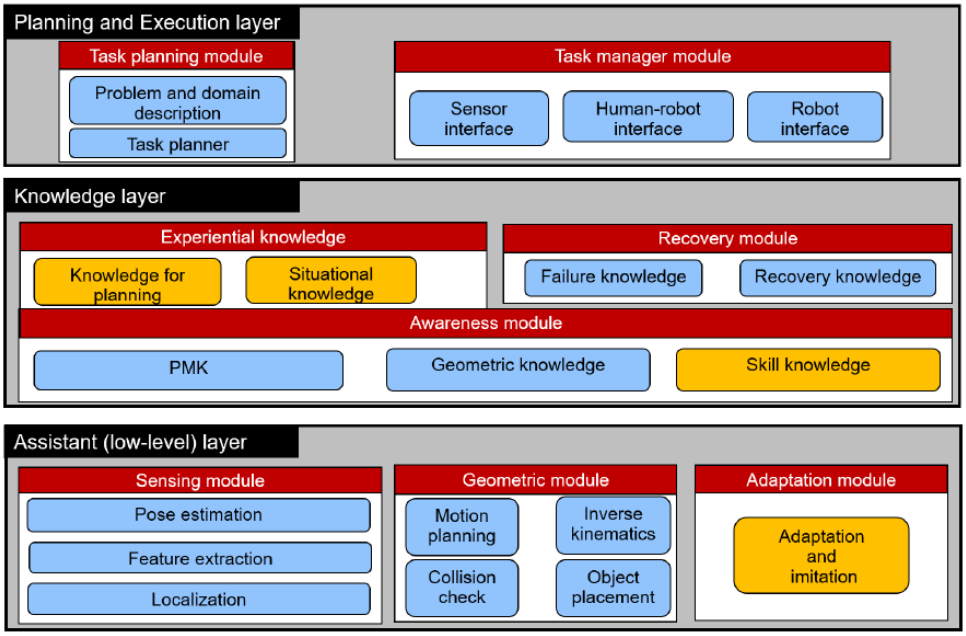


Figure 5.3: The proposed SkillMaN framework: layers and modules.

2. geometric issues, like determining if a configuration is collision-free or if an inverse kinematic solution exists for a gripper pose, which are dealt by the geometric module, as discussed in chapter 3.
3. robust issues, like the need to adapt the robot paths to the actual situations, which are dealt by the adaptation module.

The rest of the section is structured as follows. First, in Sec. 5.3 the assistant (low-level) modules used in the assistant layer will be described. Then, in Sec. 5.4 the knowledge modules in the knowledge layer used to guide the planning system and to interpret the failures in the execution phase and provide recovery strategies are presented. Moreover, the inference mechanism and data management issues are discussed. Sec. 5.5 presents the task planning and task manager modules proposed in the planning and execution layer. Finally, in Sec. 5.6 the framework flowchart illustrates how the proposed layers and their modules are integrated.

5.3 Assistant modules

This section describes the low-level modules proposed in the SkillMaN framework: sensing, geometric and adaptation module.

5.3.1 Sensing module

We extend our work presented in chapter three (PMK), by enhancing the sensing module. The sensing module integrates different types of sensors, RFID, with one-dimensional output data, and RGB-D camera, with multi-dimensional output data. The purpose of the multi-sensory integration is to cover the non-line of sight (NLOS) by using RFID technology as exploited in (Deyle, 2011), and line of sight (LOS) by using a camera. This integration allows the robot (especially the ones that have navigation capabilities) to figure out where the objects are located in an indoor environment (even if these objects are hidden, like cans inside a drawer), and the status of the objects (e.g., a can is full or empty).

a) RFID technology

An RFID technology is composed of three main parts: reader, tags and antenna. The tags, like the ones used in (Deyle et al., 2014), have a physical storage medium that allows a robot to store relevant data related to the status of the object or its relative location or spatial relationships, information that can be automatically updated from the result of the robot actions.

The use of RFID technology has appeared from the beginning of this century and most of the related works are focused on localization, like the works presented in (Deyle et al., 2014) and (Li et al., 2010). However, few efforts have been done to utilize the memory inside the tags for autonomous manipulation tasks, which requires implementing robust strategies to store and update the data in memory. Here, in this work, we make use of associated memory to store the dynamic data and update them accordingly to support the planning system by extracting the relevant information.

The purpose of using RFID in SkillMaN is:

- To partially localize the objects in the indoor environment. This allows the robot to start planning under partial information of the environment instead of discovering the entire environment which increases the computational cost of the planning process. This includes figuring out the hidden objects that the other sensors like camera can not detect. Then, the integration with other sensors like a camera can precisely recognize the objects.
- To store relevant data regarding the status of the objects in the environment, such as a *can* is full or empty, in order to adapt the manipulation behavior of the robot.

b) Camera

Using the camera and visual tags attached to the objects, the object poses are obtained (more complex untagged-based pose estimation algorithms could be used). Then, spatial relationships are extracted geometrically to understand the state of the physical world. Relationships

currently supported by the framework are *in*, *on*, *inside*, *right* and *left* as presented in (Diab et al., 2019).

The tags are used to identify the world entities and semantically link them to the the properties of each object. Specifically, the purpose of integrating the sensing module with a camera is to precisely detect the position of the objects and their IDs and assert them on the relevant ontology. Then, evaluate the spatial relations of the world entities with respect to each other and the robot (e.g., object A is located on the right side of the robot and on the left side of object B.)

5.3.2 Geometric module

The geometric module provides several services that help a planner to evaluate the feasibility of the skills, as described in chapter four. It can be summarized into four main services:

1. Inverse Kinematics (IK) used to compute the robot configurations for a given gripper pose,
2. Collision Check (CC) used to check the feasibility of a single configuration or a trajectory,
3. Motion Planning (MP) used to generate a sampled-based trajectory to be executed, and
4. Object Placement (OP) used to sample the placement region.

In SkillMaN, these services may be required in many situations in the manipulation domain, such as the case where an object is blocking the chosen configuration to grasp/place an object. This situation requires the selection of alternative feasible (or reachable) grasping poses and/or placements. Specifically, these services are used to:

1. Compute an alternative grasp or an alternative placement pose for the object.
2. Compute the IK for the new grasp or the IK for the new object placement according to the current grasp.
3. Compute a collision-free path for the new goal configuration.

These services may be required during planning to find a feasible solution, or to generate recovery strategies to recover a plan whenever a failure occurs.

5.3.3 Adaptation module

This is a module that adapts the robot motions to the actual scene based on the perceived poses of the objects in the environment. Broadly, there are two sources from where to adapt the

motions, one is from human demonstrations, the other is from collision-free motions computed by a motion planner. This is similar to what human do, i.e., we intuitively know how to perform the motion primitives, although our exact movements are only produced when we see the objects and adapt them to the scene context while we perform the skill.

The technique used for imitation motions has been the Dynamic Movement Primitives (DMPs), that implements a set of differential equations that allow describing any motion. Complex motions have long been thought to be composed of sets of primitive actions that are executed together. DMPs are a mathematical formalization of the motions of the primitives using dynamical systems theory. These dynamical systems have stable behaviors using the basic set of parameters provided by the DMP tool, although extra parameterization according to the task at hand may improve the results.

5.4 Knowledge modules

Formally, knowledge is divided into knowledge representation and inference mechanism. The former copes with how the knowledge is represented, the latter copes with how to infer the relevant knowledge. In SkillMaN, the main goal is to capture knowledge about

1. *how the planning system can be guided by the knowledge,*
2. *how the similarity of the situations can be checked,*
3. *how to manage the perception system,*
4. *whether a path is feasible or not,*
5. *how can robots perform skills and what skills are needed to achieve certain goals,*
6. *how can a situation be interpreted as a failure,* and
7. *which are the available recovery strategies for a given failure.*

The knowledge modules are first introduced and then the heterogeneous inference mechanism will be detailed at the end of the subsection.

5.4.1 Experiential knowledge module

The knowledge-based experience, or experiential knowledge, is divided into two main parts: knowledge for planning and situational knowledge.

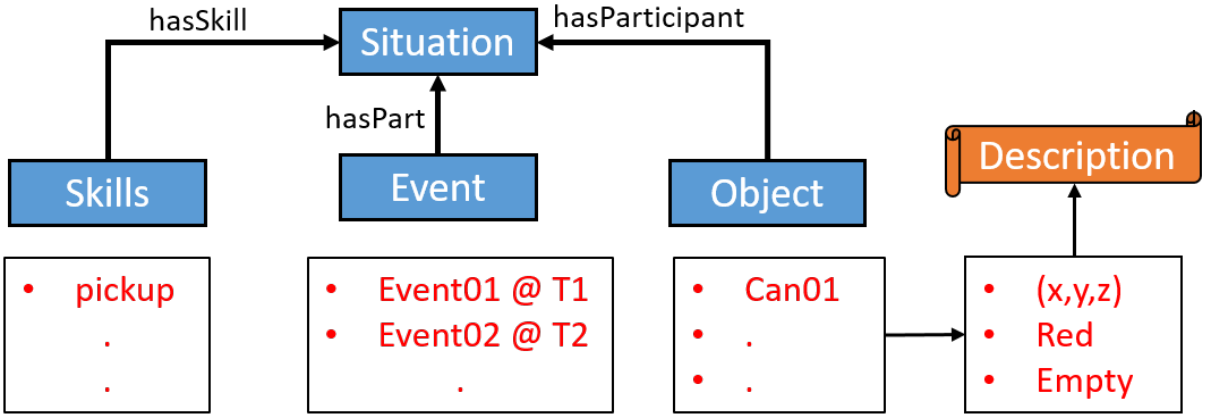


Figure 5.4: The situation modeling in SkillMaN. The blue and orange are abstract concepts, while the red referred to the instances of the classes. The multi-sensory module is used to build the description of the environment.

A) Knowledge for planning:

One of the significant requirements for the planning system is to reason about how to perform skills in semi/unstructured environments. This requires having geometric information, based on the current situation of the environmental entities, of how to manipulate the objects, e.g. which is the experience-based feasible grasp. This is what we call "geometric skills experience".

Currently, in SkillMaN, there are two sources from where to build the geometric skills experience: humans and robots. For humans, the user can build manually the geometric skill experience including the description of the task constraints (either programmed or included in an ontology). For the robots, if the robot starts exploring the way of executing an action and finds a feasible solution, it stores this solution to be later used if required. For example, let's consider the side-grasp is used for picking an object from a table and there is an obstacle occluding the path from a certain angle, several angles could be applied to explore the feasibility of the grasping configurations. Once found, the robot stores these configurations to be used in similar situations. This knowledge is required to guide the planning system especially when some motion constraints exist. In the proposed case study, several constraints have been introduced to show the importance of using geometric skills experience within a planning system as shown in Sec. 5.8.

B) Situational knowledge:

In SkillMaN, experiences are thought to be situations that provide a relational context on a set of events that occurred, and objects that were involved. This includes, e.g., *what roles an*

object plays during an skill, and *what the diagnosis is in case a failure was raised during skill execution*. Situations in our knowledge base, as described in Fig. 5.4, can be written as a tuple $< S, O, E >$, where S is the skill that was executed, O the set of objects that were involved, and E the set of events that occurred.

The representation strategy of the Descriptions and Situations ontology (Maass et al., 2007) has been followed where descriptions are used to create views on the relational context of situations. In particular, the proposed framework associates skills to the situations where the skill was executed, and exploits this information for the realization of a set of inference mechanisms.

The environment's description is used to semantically link low-level perception data with high-level knowledge, and to analyze the situation of the environment entities in order to enhance the task execution. The tagged-based sensors, i.e., RFID and camera, are used to identify the world entities and semantically link them to the properties of each object. Specifically, the purpose of the sensing module is to detect the position of the objects and their IDs and assert them on the ontology to build the description of the environment and its relevant instances following the Perception and Manipulation Kowledge (PMK) presented in chapter 3.

C) Experiential data:

Our system records sensory data over time and associates it to situations during which the data was acquired. This is mainly to capture the trajectories that were executed by the robot, and to associate them with task, environment, and execution. This is useful for machine learning applications where expressive queries can be answered at higher levels of the knowledge base, and results of such queries may serve as filter for the lower-level data to gather only the data matching a semantic situation.

In SkillMaN, two types of storage mediums are used, the memory of RFID and knowledge database. The former is used to store the dynamic data such as objects' positions and their status (e.g., a *can* is full or empty). The latter is used, beside guiding the planning system, to store the static data such as objects' features. Data management is detailed in Sec. 5.4.5

5.4.2 Awareness module

This module contains low-level knowledge related to perception, to geometric issues required for the evaluation of the actions feasibility, and to the way of executing skills.

A) Perceptual knowledge

To perceive a robot environment, different sensors are usually used. Sensors provide data

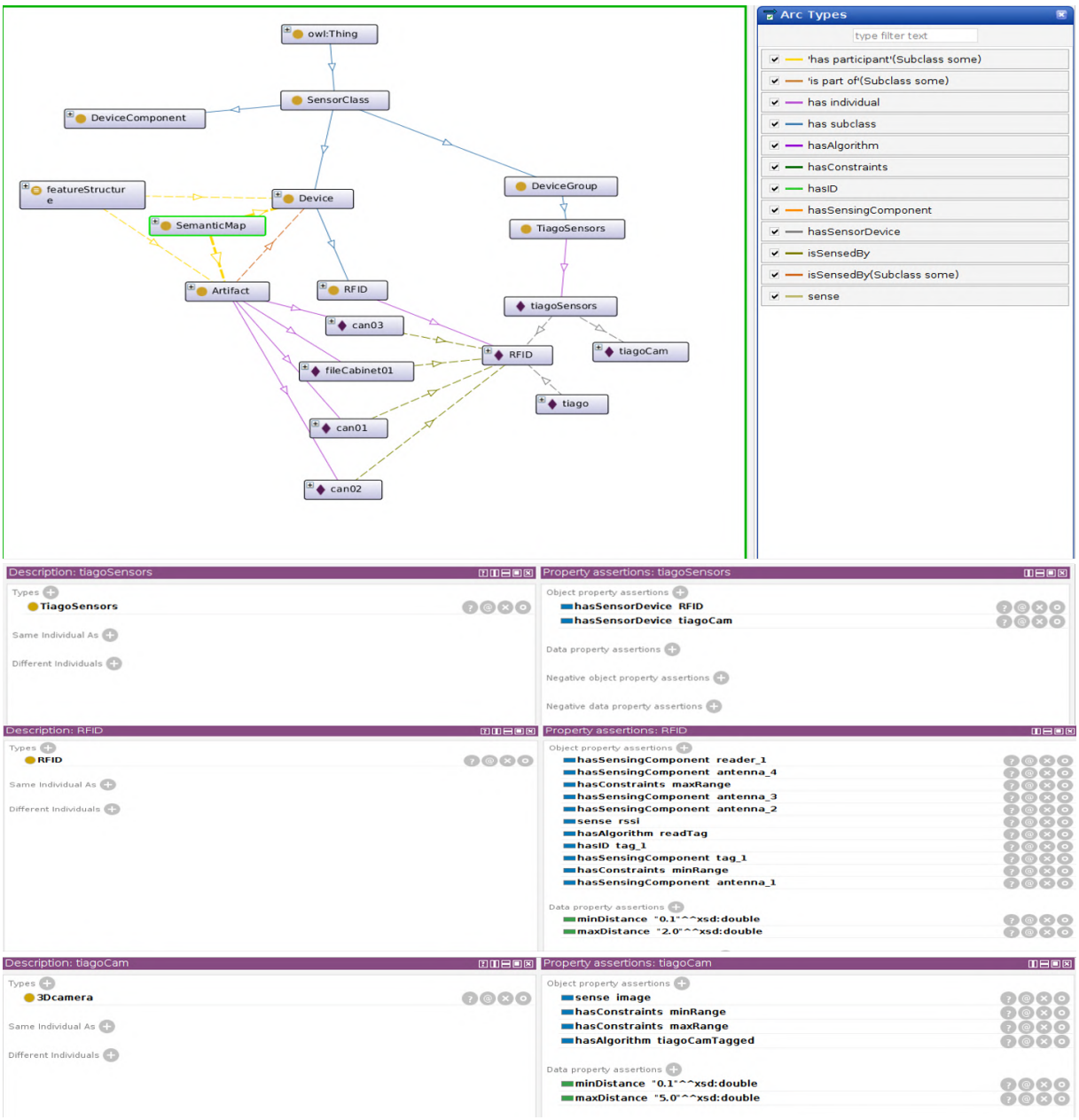


Figure 5.5: The representation of perceptual knowledge in SkillMaN.

about the environment in the form of signals (one dimension) or images (multi-dimension), and to obtain the useful features from the perceived data the suitable algorithms have to be applied, for instance to detect an object pose some pose estimation algorithms based on image features can be applied, or alternatively algorithms based on tags identification can be used. The complexity increases when the integration between the sensors exists.

Perceptual knowledge is the knowledge related to the robot sensors or to sensors associated to the environment. This knowledge is used to guide the proposed multi-sensory module. A first version of the perceptual knowledge was presented in chapter 3, where two cameras worked in parallel to perceive the table-top environment. Here, we enhance the representation of the knowledge to be capable of working with different types of sensors including RFID and camera.

The perceptual knowledge for multi-sensory integration in SkillMaN, as shown in Fig. 5.5, is represented as a tuple $< D, C, A >$ where D is a measuring device (sensor), C is the sensor constraints or limitations, and A is the corresponding algorithm to extract the features from the sensor signals.

This knowledge is responsible to answer three main questions *which are the sensors attached to the robot?, what type of data the sensors perceive and what are their limitations?, how to extract the relevant data?*. To answer the first question, a description of the sensors is proposed to make the robot understand which are the group of sensors it has. Moreover a description of the components of each sensor, like the tags, antennas and reader of the RFID is included. To answer the second question, a description of the perceptual features is proposed to clarify to the robot which type of data (i.e, one or multi dimensions) the sensors are perceiving, and what are the constraints or limitations of each sensor. To answer the third question, a method to call the corresponding algorithms is proposed to extract the relevant data.

Using Description Logic (DL, (Baader et al., 2017)), the knowledge of RFID sensors is expressed as:

$$
\begin{aligned}
&RFIDKnowledge :- \\
&\exists hasSuperclass(RFID, Sensor) \\
&\wedge \exists \textit{Sense}(RFID, RSSI) \\
&\wedge \exists \textit{hasSensingComponents}(RFID, Tag) \\
&\wedge \exists \textit{hasSensingComponents}(RFID, Reader) \\
&\wedge \exists \textit{hasSensingComponents}(RFID, Antenna) \\
&\wedge \exists \textit{hasID}(RFID, taggedID) \\
&\wedge \exists \textit{hasConstraints}(RFID, minRange) \\
&\wedge \exists \textit{hasConstraints}(RFID, maxRange) \\
&\wedge \exists hasAlgorithm(RFID, readTag)
\end{aligned}
$$

And the knowledge of camera is expressed as:

$$
\begin{aligned}
&CameraKnowledge:-\\
&\exists hasSuperclass(Camera, Sensor)\\
&\wedge\exists \textit{Sense}(Camera, Image)\\
&\wedge\exists \textit{hasConstraints}(Camera, minRange)\\
&\wedge\exists \textit{hasConstraints}(Camera, maxRange)\\
&\wedge\exists hasAlgorithm(Camera, tiagoCam)
\end{aligned}
$$

B) Geometric knowledge

The geometric knowledge has a structure for sequential access to the geometric services in the assistant layer. The main advantage of this ontology is that, instead of calling the module manually from the task and motion planning client, the robot can query over the knowledge to retrieve the sequence of processes required to execute such actions in an automatic way.

C) Skill knowledge

A skill, in SkillMaN, is a description of what the robot can do. The SkillMaN provides some methods of teaching new skills to the robot inspired by the work presented in (Munawar et al., 2018):

1. Primitive skills consist of a sequential list of atomic actions, which refers to a single action or gesture, including its preconditions and effects. For example, an *openDrawer* skill is composed of the sequence of actions: move to the handle position, close the gripper, and finally pull the drawer.

2. Rule-based skills consist of a set of “if A then B rules" to issue appropriate gestures according to sensors outcome.

Both methods, however, cannot be executed on their own. They require a structure, such as a workflow, that contains the abstract steps that are usually required for task execution. This structure is described at a symbolic level and grounded to be attached to each skill using the assistant layer. The main difference between the both aforementioned methods is a perception-based conditional node in the structure. That means the structure includes some branches of a given value that should be sensed.

5.4.3 Recovery module

Knowledge for recovery is a module that provides an interpretation of the failures that occur. In SkillMaN, an interpretation failure ontology described in chapter 4 covering several sources of failures, is used both during planning and execution. It offers recovery strategies for:

1. Geometric failures, that may appear when e.g. the robot can not reach to grasp/place an object, there is no collision-free path or there is no feasible Inverse Kinematic (IK) solution;
2. Hardware related failures that may appear when e.g. the robot in a real environment requires to be re-calibrated (gripper or arm), or it is sent to a non-reachable configuration;
3. Software agent related failures, that may appear when e.g. the robot has software components that fail like when an algorithm is not able to extract the proper features.

5.4.4 Heterogeneous inference mechanism

This section presents a heterogeneous way of reasoning that includes symbolic reasoning over the knowledge module and geometric reasoning. The former includes filtering the situation from the database, situation similarity check, skill reasoning, semantic reasoning regarding the environment and its entities, manipulation constraints and perception. The latter includes the geometric reasoning to check the feasibility of the generated skills. They are discussed below.

Symbolic reasoning

A.1) Filtering situation:

Filtering situation is a process of finding those situations that *satisfies* a skill description. It means the robot has to detect the situations that use a specific skill in their description, e.g. using a Prolog predicate (WIELEMAKER et al., 2012) the reasoning on *"which are the situations that contain a certain skill?"* is:

```
?- filterSituation( hasSkill(Situation, Skill) ),
?- filterSituation( hasParticipant(Situation, Object) ),
?- filterSituation( hasPart(Situation, Event) ).
Situation=[Skill, Object, Event].
```

A.2) Situation similarity check:

The similarity of situation scenes is computed using taxonomic information from the situational ontology together with information about what makes up the compared entities. Note that scenes are compound entities – that is, a scene has objects and agents as participants.

Figure 5.6: An example of similarity check between two scenes. This example is also a part of the experimental scenes of scenario one in Sec. 5.8 (storage task).

Objects and agents themselves are compound entities; an object or agent may have other objects as parts. Also, the description of an agent includes the skills to be executed, geometric-skills experience, and agent goal.

Entities that are considered simple – the parts of objects or agents – are compared using Wu-Palmer similarity (Wu and Palmer, 1994), although for numerical stability reasons, the logarithm of this similarity score is used here, i.e., for two individuals x and y:

$$\text{Sim(x, y)} = \log \frac{\text{depth (lca C(x), C(y))}}{0.5(\text{depth C(x)}) + \text{depth C(y)})} \tag{5.1}$$

where $C(x)$ is the class to which individual x belongs, $depth(A)$ is the depth of class A in a taxonomy, and $lca(A, B)$ is the lowest common ancestor of classes A, B in that taxonomy. The intuition behind Wu-Palmer similarity is that similar classes should be close to each other in the taxonomy.

To compare individuals x, y that are compound entities, their parts are matched such that for every part x_p of x, we find the part y_p of y that maximizes $Sim(x_p, y_p)$. Then, the sum of the similarity scores obtained from these matching is added to $Sim(x, y)$. The intuition here is that we want to have the similarity of complex objects such as situations or scenarios to depend on the nature of those scenarios as well as their participants.

We only compare the "tree" of part-hood relations for efficiency reasons. In principle, there may be many stored scenes one could compare the current situation to, and filtering out most of them so that only a few relevant candidates remain. Once some candidate similar scenes are selected, the more intensive procedures of adapting robot motion from the stored scene to the current one can be used to ascertain the usefulness of the stored experience for the current task.

For example, as shown in Fig.5.6, if the robot has successfully planned the picking of the black can in scene a), this situation, that contains the objects, geometric-skills experience, and skills, is stored in the ontology database. In b), and after checking the similarity of the scenes, the robot uses the same geometric skills (grasp from the top), especially because the robot has the same goal (i.e., to place the can in a drawer). More details about this example are mentioned in the experimental section.

A3.) Skill reasoning:

Symbolic skill reasoning using rule-based logic is used

1. Load the appropriate PDDL domain file, i.e., The purpose of the inference in this level is to answer the question to any task planner, *"what is the world description?"*, which using a Prolog predicate is:

```
?- loadPDDL( hasfile(PDDLdomainFile, fileID) ).
File = PDDL domain file.
```

2. Load the appropriate PDDL problem file that fits the current situation, i.e., the question *"what is the problem?"*, should be answered, which using a Prolog predicate is:

```
?- loadPDDL( hasfile(PDDLproblemFile, fileID) ).
File = PDDL problem file.
```

3. To check the satisfaction of the constraints of a situation, i.e., the question *"Do preconditions of a skill hold?"*, which using a Prolog predicate is:

```
?- SkillPrecondiction( hasStatus(Object, Status) ),
?- SkillPrecondiction( isReadFrom(Status, Sensor) ).
?- SkillPrecondiction( satisfies(Status, SkillPrecond) ).
SkillPrecondiction = Boolean value (True or False).
```

This predicate returns a Boolean value which identifies whether the skill preconditions are satisfied. It is used in the scenario described in Sec. 5.8 (storage task), where the RFID tag memory is used to know the status of the drawer (i.e., open or closed).

A4.) Semantic reasoning:

Beside the aforementioned reasoning process, an expressive inference process is proposed to identify the hidden knowledge and increase the robots capabilities, as used in chapter 3 and 4. The semantic reasoning obtains the knowledge that the robot requires to manipulate the objects in the environment (e.g, *what are the fixed and manipulatable objects?*), reasoning related to sensing (e.g, *what is the corresponding algorithm?*), task planning (e.g, *what is the state of the objects in the current scene?*), and motion planning (e.g, *which are the interaction parameters that hold?*).

B) Geometric reasoning

The main role of geometric reasoning is to evaluate geometric conditions of symbolic skills. Two main geometric reasoning processes are provided:

B.1) Reachability Reasoning:

A robot can transit to a pose if it has a valid goal configuration. This is inferred by calling an Inverse Kinematic (IK) module and evaluating whether the IK solution is collision-free. The first found collision-free IK solution is returned, and, if any, the associated pose. Failure may occur if either no IK solution exists or if no collision-free IK solution exists.

B.2) Spatial Reasoning:

We use this module to find a placement for an object within a given region. For the desired object, a pose is sampled that lies in the surface region, and is checked for collisions with other objects, and whether there is enough space to place the object. If the sampled pose is feasible, it is returned. Otherwise, another sample will be tried. If all attempted samples are infeasible, the reasoner reports failure, which can be due to a collision with the objects, or because there is not enough space for the object.

Example of the Prolog predicates for the geometric reasoning are:

"Is the path toward a goal configurations reachable?":

```
?- testReachability( hasAlgorithm(IKModule, IKAlgorithm) ).
Algorithm = Call IK service.
```

"Is the path toward a goal configurations collision-free?":

```
?- CollisionCheck( hasAlgorithm(CCModule, CCAlgorithm) ).
Algorithm = Call collision check service.
```

Also, these can be a part of a general workflow to call IK, collision check and motion planning in a sequence as follow:

```
?- testReachability( hasAlgorithm(IKModule, IKAlgorithm) ),
?- CollisionCheck( hasAlgorithm(CCModule, CCAlgorithm) ),
?- MotionPlanning( hasAlgorithm(MPModule, MPAlgorithm) ).

Algorithm = Call [IK, CC, MP].
```

5.4.5 Data management

In this section, different sources to retrieve/reason about the data related to the environment, its entities and the way of manipulation are introduced. We divide the knowledge into static

and dynamic knowledge. Static knowledge is used to describe the static data related to the environment, its entities such as the color and dimension of an object. Dynamic knowledge is used to describe the dynamic data such as spatial relations, positions of the objects and the way of manipulation based on the objects situation (e.g, the literal grasping pose of a can from a drawer is from the top).

The management of the information related to the manipulation of objects (their properties and ways to be manipulated), requires the integration of the data stored in the RFID tags memory and that stored in an Database (DB) and knowledge in ontology form.

A) Static knowledge:

DB and ontologies are used to store the static data. DB is particularly used to store the skill-based experience with their trajectories to be called when the situation fits. The ontology is used to structure the abstract relations that semantically describe the environment and its entities in a hierarchical way.

B) Dynamic knowledge:

RFID tags memory play a significant role to store dynamic/partial data. Relevant data like spatial relations can be smoothly stored in the memory. The framework has capability of updating the relevant data based on the robot result of actions. For example, the relevant dynamic data could be the status of a *can* (full or empty) and if the robot has done a skill to serve a *can* in a *cup*, based on the result of this action, the status can be automatically updated. Moreover, asserting some information related to a new instance in the environment like a *cup01* belongs to a category *cup* in the ontology, allows the instance to take the same taxonomy of the *cup* super class.

5.5 Planning and execution modules

In this section, the module of task planning and the task manager module responsible for managing the tasks are described. Also, the framework flowchart showing the linkage of the proposed modules is introduced.

```
(:skill openDrawer
        :parameters (?rob - robot ?cont - containerAll ?st - status ?st2 - status)
        :precondition (and (= ?st closed) (status ?cont ?st)
                           (not (infeasible drawer2 drawer3 table))
                           (holding ?rob ?cont fileCabinet)
                           (forall (?ob2)
                           (not (isCritReach ?ob2 ?cont fileCabinet) ) ) )
        :effect (and (not (status ?cont ?st) )
                     (status ?cont ?st2) (not (holding ?rob ?cont fileCabinet) )
                     (arm-empty ?rob) ) )
```

Figure 5.7: *openDrawer* skill representation in PDDL. It contains parameters (the description of the skill), the skill preconditions (drawer should be closed) and its effect (expected state).

5.5.1 Task planning module

The symbolic planner is responsible to observe the current state of the system, understand the goal and generate a sequence of skills to achieve the goal. The initial state of the world is extracted by the “initial state extractor or general perception". The initial state is the truth about the world and is always detected before computing a plan.

Generally, the planning systems require a description of the environment that could be complete or partial. In SkillMaN, the planning starts with only defining the target object instead of discovering the entire environment. That means some objects are not considered as participants of the robot working space. This planning process is considered as a planning under partial information. The main advantage is that it reduces the computational cost of the planning process. Here, it is done by using the integration of RFID and vision sensors.

Computing poses of all objects is not enough, we also need to define their state, e.g. if a *can* is filled or not, which can be done by the perception service using the RFID memory associated with the tags.

A plan is conceptually a high-level abstraction for going from a given initial state to a goal state. It consists of a sequence of skills that are defined, learned, or computed by a symbolic planner.

As described in Sec. 5.4.2, skills refer to an executable set of actions which require motion. In SkillMaN, the following essential skills are proposed to deal with several scenarios in indoor robotic manipulation:

1. *pickUp:* a skill done by the robot to move an attached object from one pose to another one.

2. *putDown:* a skill done by the robot to place an attached object to a certain pose.

3. *openDrawer:* a skill done by the robot to move an articulated object with a prismatic joint like to open a drawer or box-like container.

As illustrated in Fig. 5.7, the skills in the skill database can be defined as actions in PDDL for a particular domain, in which the preconditions and effects of the skills are described. The domain and problem PDDL files are labeled in the skill knowledge. A planning query is given as a PDDL problem, in which the goal and initial states are described with semantic labels. Extended semantic annotations enable the linking between actions and problems using semantic resolution, which allows to overcome the closed-world assumption of the PDDL domain by automatically updating the states of the world and the potential actions that the robot can perform. The semantics, actions, and required relationships are defined in PDDL domain files, therefore, a suitable domain file must be defined explicitly by the user to generate a plan for the problem. The output of the planner is a sequence of skills that is executed at runtime, which might require the planner to generate an alternative plan in case of a failure.

5.5.2 Task manager module

In this module, the perception, planning (symbolic and geometric), adaptation and knowledge are combined. After generating the scene using the initial state extractor with the guidance of perceptual knowledge, and computing the sequence of skills, the geometric module is used to compute the collision-free path of each skill. Moreover, the knowledge-based reasoning is used in this module to check the similarity of the situations with the ones stored in the database.

The planner usually generates atomic actions or skills for the robots, although atomic actions can also be generated for humans or sensors. For example, one of the planning actions could be sense the environment. That means that the perception system uses a sensor and feeds back the planner with the relevant information, or asks a human operator for assistance. This module provides the interface for passing the planned-based skills for the robot, sensors and humans. More details about this module is described in Sec. 5.7.2

5.6 Framework flowchart

The flowchart shown in Fig.5.8 illustrates how the proposed modules in the framework are integrated for such tasks. Firstly, the robot needs to read which are the initial and goal states. The initial state can be extracted using the perception system. The exact location of the objects can be completely detected if they are, for instance, in the field of view of a camera, or only partially located by using the RFID. Then, the task planner computes the sequence of skills to be

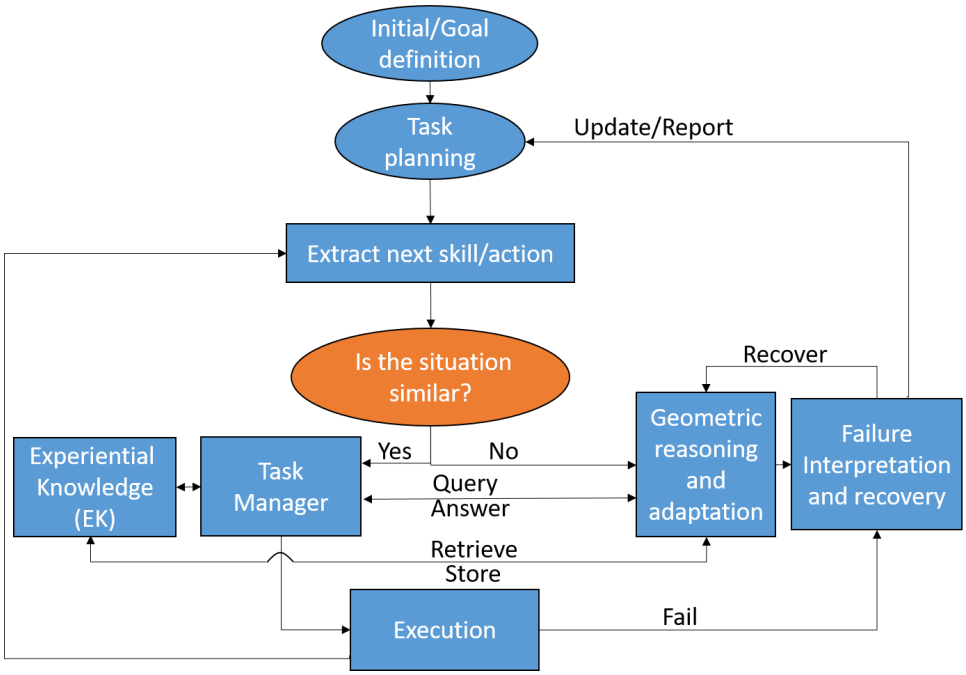


Figure 5.8: Flowchart of the SkillMaN framework.

executed, what is called *general plan*. These sequence of skills is obtained at a symbolic level, without any geometric considerations.

The robot starts to follow the general plan without any geometric consideration and, based on the perception outcomes and robot goals, it semantically reasons on which situations stored in experiential knowledge use the same skill (filtration process). In the case of similarity, the task manager determines how to execute the skill by querying the adaptation module, that retrieves the information from the experiential knowledge.

If the adapted motion is not collision-free or if no similarity was found, then the robot queries the geometric reasoning modules, i.e., inverse kinematic, collision check, motion planning, and object placement, to generate a motion for the skill (also, if necessary, with the assistance of the experiential knowledge to provide relevant geometric information like the best grasping configuration to be used). After the motion is generated, it is stored in the Database (DB) as experience to be called whenever needed.

5.7 Implementation and set-up

5.7.1 Implementation tools

A) Perception

The C++ library ar–track–alvar (http://wiki.ros.org/ar_track_alvar) has been used to detect the object pose and ID. Moreover, the C++ library ThingMagic Mercury API (http://www.thingmagic.com/manuals-firmware) of RFID technology has been used to detect the objects, including the hidden ones, and to store the relevant dynamic information. Some services are implemented to read the tagID, read the data from memory and write/update the data on the memory. These IDs are asserted in the knowledge to extract a semantic description of the object. All the transformations of the objects and camera are calculated with respect to the world frame.

B) Planning and adaptation

A planning system consists of two main phases: task planning and motion planning. The first is implemented using the Fast Forward (FF) task planner to generate a sequence of actions. The latter is implemented using The Kautham Project (Rosell et al., 2014). The Kautham Project is a C++ based open-source tool for motion planning, that enables to plan under geometric and kinodynamic constraints. It uses the Open Motion Planning Library (OMPL) (Sucan et al., 2012) as a core set of sampling-based planning algorithms. In this work, the RRT-Connect motion planner is used to generate a path between two configurations.

The main technique onto which the imitation motions have been implemented is the DMP (Ijspeert et al., 2002). The DMP experiments are performed first in simulation and afterwards using the real robot. The experiment consisted in learning by recording the execution of the planned-base motion and then changing the initial and final points to see how the planned gestures are imitated.

An interface has been implemented to command the arm to perform such motions that require imitation gestures. This interface can be divided into three main parts, the data acquisition process, the DMPs generation, and the execution of the motion. The data acquisition is performed by recording the motion. The DMPs generation is done using the motion recorded as input to learn how to perform the DMP primitive in a new situation. The execution of the motion uses the initial configuration and goal state required to adapt the motion in similar situations.

The integration between the planning and adaptation tools is done automatically together in the preparation phase.

In SkillMaN, the proposed abstract primitives are described:

1. *releaseGripper:* an atomic action used to open the gripper.
2. *closeGripper:* an atomic action used to close the gripper.
3. *pick-place:* an skill that contains the atomic actions move, hold and put-down; it is used for transferring the objects between two locations.
4. *openDrawer:* is an skill that contains the atomic actions move, hold and pull; it is used for opening/closing the drawers.
5. *serving:* is a skill that contains the actions move and pour; it is used for serving the beverages to a customer.

The motions that are adapted are initially either computed by the motion planner, e.g, in case 3 and 4, or copied from human demonstrations, e.g, in case of 5. In case of 1 and 2, the motion of closing and opening the gripper is predefined.

These abstract primitives correspond to the basic functions of the robot manipulator, which can be implemented in many different ways. Our way of implementing such primitives is at the lowest motor control level. The focus of our work is, however, not a specific implementation, but rather we would like to propose a way to combine them to seamlessly perform skills.

C) Knowledge

The knowledge is designed using ontology web language (OWL) using the Protégé ontology editor (http://protege.stanford.edu/). Ontology instances can be asserted using information processed from low-level sensory data.

Queries over the knowledge to reason or check the similarity are based on SWI-Prolog and its Semantic Web library which serves for loading and accessing ontologies represented in the OWL using Prolog predicates. A ROS (Robot operating System) interface has been implemented in order to facilitate the query-answer process as a client-service communication.

The PMK approach, as presented in chapter 3 is used in this work. It is explicitly implemented to enhance Task and Motion Planning (TAMP) capabilities in the manipulation domain. It is integrated with the multi-sensory module allowing the instances to be asserted to the ontology using information processed from low-level sensory data.

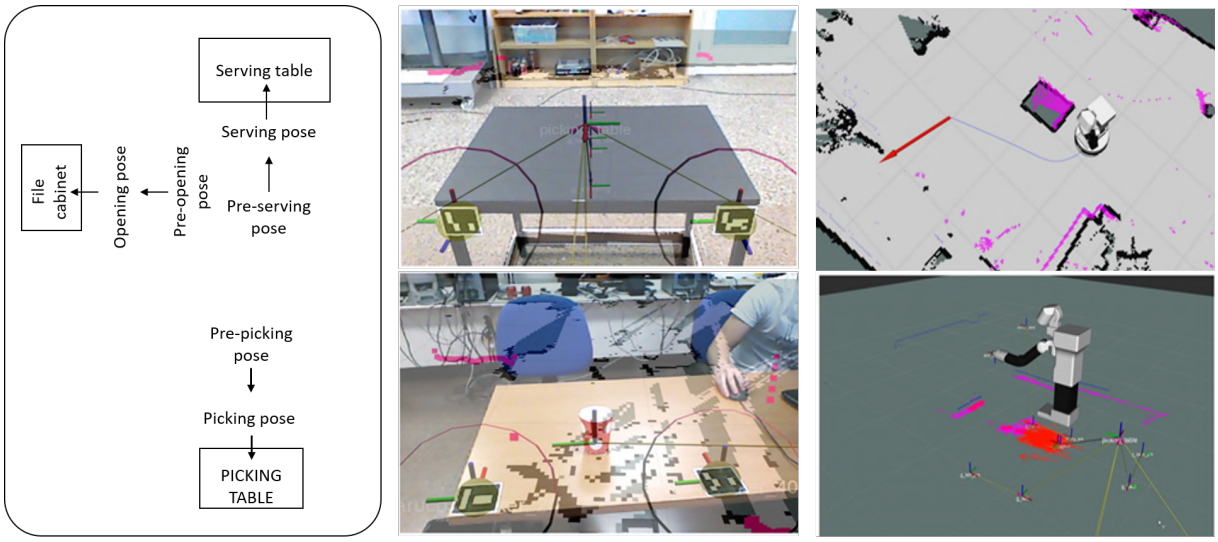


Figure 5.9: Navigation experiment: (a) the plan view of the indoor environment, (b) the real scene of how the robot detects the tables used in the indoor environment, and (c) path planning, obstacle avoidance capabilities, and navigation poses on the map.

D) Navigation and mapping

Fig. 5.9 describes the navigation strategy proposed in SkillMaN. In (a), the plan view of the indoor environment has been shown. By using the mobile capacity of TIAGo, it plans toward the navigation position of the objects (e.g, TIAGo navigate toward the picking and serving tables) until it detects the labels attached to them, as shown in (b). During the navigation, TIAGo has the capabilities of path planning with obstacle avoidance and localization of the objects in the map, as shown in (c). All the objects in the environment are localized with respect to the reference frame.

D.1) Navigation: TIAGo has autonomous navigation functionalities implemented using the ROS 2D navigation stack (http://wiki.ros.org/navigation). This package is one of the most commonly used to implement mapping and autonomous navigation solutions in robots running on ROS. It takes in information from odometry and sensor streams and outputs velocity commands to send to the mobile base. This navigation software is composed of several different ROS nodes, services and topics that are able to perform SLAM (Durrant-Whyte and Bailey, 2006). Using the information stored on the map and the data of its surroundings provided by different sensors, this package is capable of computing a suitable path to lead the robot to a certain goal position without hitting any obstacle.

D.2) Mapping: The mapping and pose generation process starts by creating the occupancy

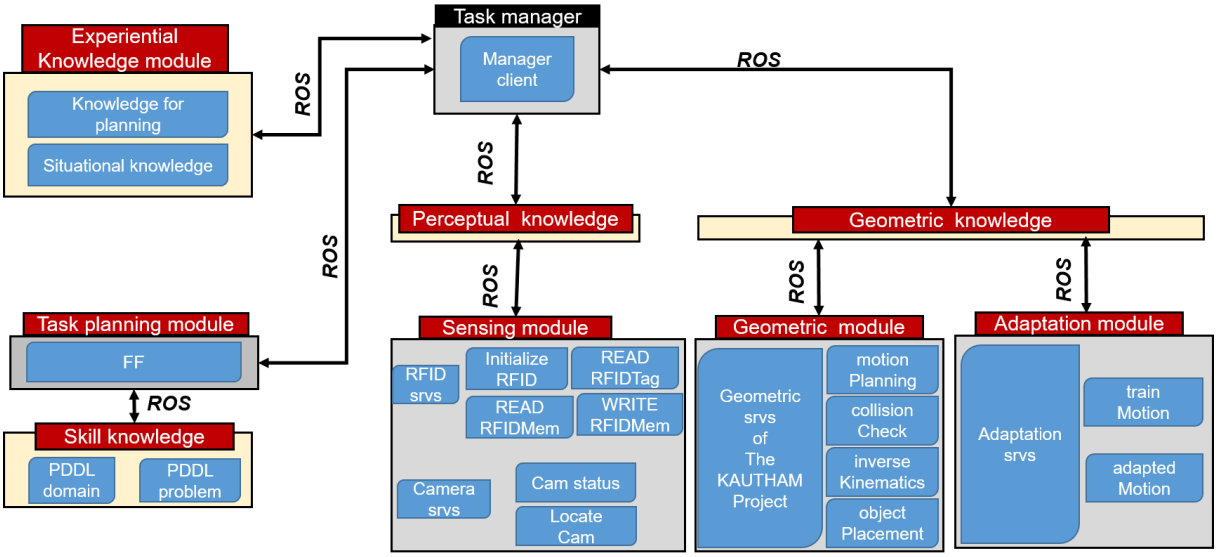


Figure 5.10: Task management and the communication with the ROS-based services from symbolic and low-level modules.

grid map of the environment of the robot. To obtain it, the *gmapping* (http://wiki.ros.org/gmapping) package installed in the robot has been exploited. This map is necessary for the navigation to successfully move through the room avoiding any collision.

D.3) Localization: Localization is achieved by working with the amcl package (http://wiki.ros.org/amcl). This package is a probabilistic localization system for a robot moving in 2D. It implements the adaptive Monte Carlo localization approach (MCL), which uses a particle filter to track the pose of a robot against a known map. MCL generates a cloud of particles which represent the possible states of the robot distribution. Each particle represents a possible pose and orientation of the robot on the map.

5.7.2 Task manager algorithm

The SkillMaN is not implemented for a specific task, it is quite general and it accepts several tasks in indoor environments with the consideration of some changes regarding the description of the environment, as discussed in Sec. 5.9.3. All the modules mentioned in Fig. 5.3, and the provided services of each module as described in Fig. 5.10, are used by the task manager. The task manager is responsible to call these services in order to autonomously execute the tasks, as described in Algorithm 1.

The sensing module is managed through the perceptual knowledge, the following service is

Algorithm 1: $taskManager$

```
1  initialState ← runPerception (RFID) //  run RFID sensor to perceive the environment using the
   perceptual knowledge
2  while Task goal not delivered do
3      D = loadPDDL // load the domain and problem files from skill knowledge
4      P = FF(D) // plan at symbolic level to compute the sequence of skills
5      skillName ← firstAction(P)
6      while skillName do
7          objects, poses ← runPerception(Camera) // perceive the actual state of the
           environment according to the skill
8          Y = skillName, objects // store the current situation.
9          F =filterSituation(skillName) // return a set of situations that use the same skill
10         S = similarityCheck(F, Y) // return the situation which is most similar in the similarity
           check, if any
11         if S! = emptySet then
12             Mot ← loadMotion (S)
13             expKnow ← experientialKnow // return the geometric-skills experience
14             AdaptedMot ← adaptMotion(Mot, expKnow) // adapt motion into the current
               situation
15             C = checkFeasibility(AdaptedMot) // verify the feasibility of the adapted
               motion
16             if AdaptedMot = feasible then
17                 execute(AdaptedMot)
18                 store[AdaptedMot, skillName, Y] // store the executed motion, skill and the
                   current scene situation
19         if S = emptySet or C = infeasible then
20             Mot = generateMotion(skillName) // generate a collision-free motion for a
               new situation
21             if Mot = feasible then
22                 execute(Mot)
23                 store[Mot, skillName, Y]
24         skillName ← nextAction(P)
```

used for this purpose:

- *runPerception* used to select the corresponding algorithm(s) associated to the available sensors.

The sensing services are used to provide the initial state of the environment to the planner or whenever required, using RFID sensor and cameras.

The RFID sensor has four main services:

1. *InitializeRFID* used to set up the RFID system (i.e., reader, antennas and tags),
2. *ReadRFIDTag* used to read the RFID tags ID associated to the entities,
3. *ReadRFIDMem* used to read the dynamic data stored in tags' memory, and
4. *WriteRFIDMem* used to update the tags' memory.

The camera has two main services:

1. *Cam status* used to initialize the camera, and to input (from either a human guidance process or a motion planner) the motion to be adapted, and
2. *LocateCam* used to estimate the objects poses and their IDs.

In the planning phase, two services have been used to call the the heuristic-based task planner FF (Fast Forward) and to load the domain and problem files:

1. *loadPDDL* used to automatically load the PDDL domain and problem files, as described in Sec. 5.4.4
2. *FF* used to automatically compute symbolically the sequence of skills to be executed.

In the analysis of each action the solution plan, with a guidance from knowledge modules, the following services are used:

1. *filterSituation* used to filter the situations that include a specific skill in the database,
2. *similarityCheck* used to compare the current situation with the others stored in the database,

3. *experientialKnow* used to provide geometric skills, based on the robot experience, e.g. the type of grasp according to the current situation.

The geometric services are manged through the geometric knowledge by calling following service:

- *generateMotion* used to compute the initial and goal configurations (according to the action/skill to be performed and the reachability and spatial reasoning predicates proposed in Sec. 5.4.4), and the collision-free path between them using the *motionPlanning* service.

The geometric services are used to combine the geometric module with a symbolic planning level to guarantee the feasibility of the planned skills. This module has four main services, provided by The Kautham Project:

1. *motionPlanning* used to compute a collision-free path,
2. *collisionCheck* used to verify whether a robot configuration is collision-free or an object at a given pose is not interfering with others,
3. *inverseKinematics* used to compute the robot configurations for a given desired pose of the end-effector,
4. *objectPlacement* used to sample/check the availability of placement locations for the objects.

The adaptation services are used to imitate/adapt the motion of each skill to be executed in such situations. There are two services managed through the geometric knowledge:

1. *trainMotion* used to input (from either a human guidance process or a motion planner) the motion to be adapted and returns the weights used to shape the this motion, and
2. *adaptMotion* used to compute the initial and goal configurations (according to the action/skill to be performed and the reachability and spatial reasoning predicates proposed in Sec. 5.4.4), and to adapt the trained motion is verified with the *trainMotion* service.

This interface is established based on ROS (Robot Operating System) service-client communication.

5.7.3 Experimental set-up

The experiment has been done at IOC lab and it is composed of:

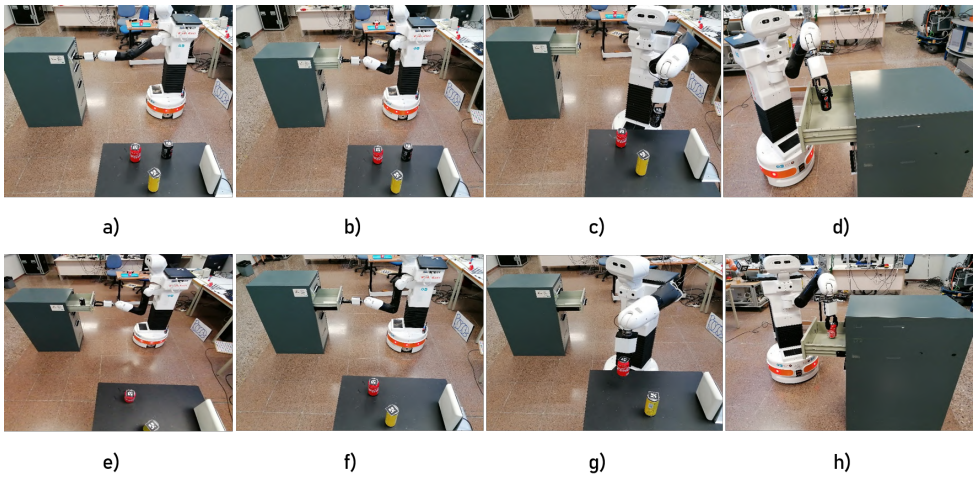


Figure 5.11: a) the robot plans how to open the first drawer; b) the robot executes the openDrawer skill; c) the robot plans how to pick the black can (based on its status, here it is empty) with the help of experiential knowledge about what is the best grasp to place it in the first drawer; d) the robot executes the place skill; e) the robot executes the close action using the rule-based skill; f) the robot checks the similarity of the current situations, it finds the same skill has been used with the same object (a drawer in the file cabinet), then executes the skill with the same motion used to open the first drawer; g) the robot plans how to pick the red can (based on its status, here it is full) with the help of experiential knowledge about what is the best grasp to place it in the second drawer; h) then, the robot executes the place skill. Video URL: https://www.youtube.com/watch?v=bTmWAkjC93c

1. The TIAGo robot.
2. A file cabinet with four drawers.
3. A storage table that contains the objects (*cans*).
4. Several *cans* that may be full or empty.
5. A serving table that contains a *cup* on a *tray* where the robot must pour the contents of a can to a customer.
6. A perception system with camera and an RFID sensor that includes:
   - A reader that has the capacity of reading four antennas, distributed around the lab, and let the robot determine the region where the objects are located.
   - The tags which have a unique ID and a memory with a space of 64 characters.

Figure 5.12: a) the robot checks the similarity of the current situation, it finds the same skill has been used with the same object (i.e., a drawer in the file cabinet), then adapts the skill with the same motion used in the database; b) the robot figures out the top-grasp is not feasible for pouring action, the top of the file cabinet is used as a placement room to change the grasp type; c) the robot changes the grasp type from the top-grasp to the side-grasp; d) the robot serves the contents of the can in the cup to a customer, the serve motion is adapted from the experience, according to the current pose of the robot and location of the cup. Video URL: https://www.youtube.com/watch?v=bTmWAkjC93c

5.8 Experimental scenarios

Before describing the proposed scenarios, some assumptions should be taken into account.

1. All the object models are defined in the knowledge database.
2. All atomic actions/skills to be used are described with their preconditions and effects in knowledge database.
3. All the RFID's antennas are distributed in the environment in a way that avoids the interaction between the signals received from each one. That means that each antenna only receives the information of the tags located in its coverage region.
4. All environmental entities are labeled with either RFID or vision-based tags.
5. All the features received from the multi-sensory perception system (RFID and camera) are reliable enough.

5.8.1 Scenario one: Storage task

Fig. 5.11 shows a sequence of snapshots of the storage *cans* task. The task is to classify the cans on the table to store them in the drawers based on their status. Firstly, the robot checks the status of the selected *can* by reading this information from RFID memory. After computing the symbolic plan, the robot can apply the skill *openDrawer* (the first action of the symbolic plan)

to the corresponding drawer of the file cabinet, as shown in Fig. 5.11 a-b. Then, to generate a collision-free path, the motion planner has been called. Then, with guidance from semantic knowledge and the experiential knowledge (if geometric experience exist), the robot can reason about how to apply the *pickUp* skill to the *can* from the table and how to *putDown* it inside the drawer and close the drawer, as shown in Fig. 5.11 c-e. After similarity check process of the current situation i.e., comparing it with the ones that have a similar description stored in the database, the imitation process is used to imitate those skills to be applied for the other *cans*, as shown in Fig. 5.11-f. The second *can* is full and the corresponding drawer where to be stored is the second one, shown in Fig. 5.11 g-h. The perception system is used to check if the preconditions of the actions are satisfied or not. For example, to apply the *openDrawer* skill in the second drawer, the first one should be closed to allow the robot to *putDown* the selected *can* correctly in the drawer.

5.8.2 Scenario two: Serving task

Fig. 5.12 shows a sequence of snapshots of the serving *can* task. The task to serve the contents of the *can* inside a *cup* on the *tray*. The robot can not apply the *pouring* skill with the grasping pose used to pick it up from the drawer (which is top-grasp). The robot needs to temporally place the can to change the grasp from the top to side grasping configuration. The top surface of the drawer is used as a free placement room for this sub-task. Then the robot is able to serve the *can*. Moreover, if the position of the *cup* is changed, the robot will be able to adapt the motion with the new position.

Table 5.1: Test the skill *openDrawer, pickUp* and *serving* using adaptation method vs the planning system with and without experiential knowledge.

Skill	**Parameter**	**Adaptation**	**Planning**	
			With Exp.know	**Without Exp.know**
$openDrawer$	S – success rate	100	100	50
	T – Avg. time (sec.)	4.5	11.5	39.5
$pickUpFromDrawer$	S – success rate	100	100	50
	T – Avg. time (sec.)	12.5	33.5	58.5
$serving$	S – success rate	100	–	–
	T – Avg. time (sec.)	12	–	–

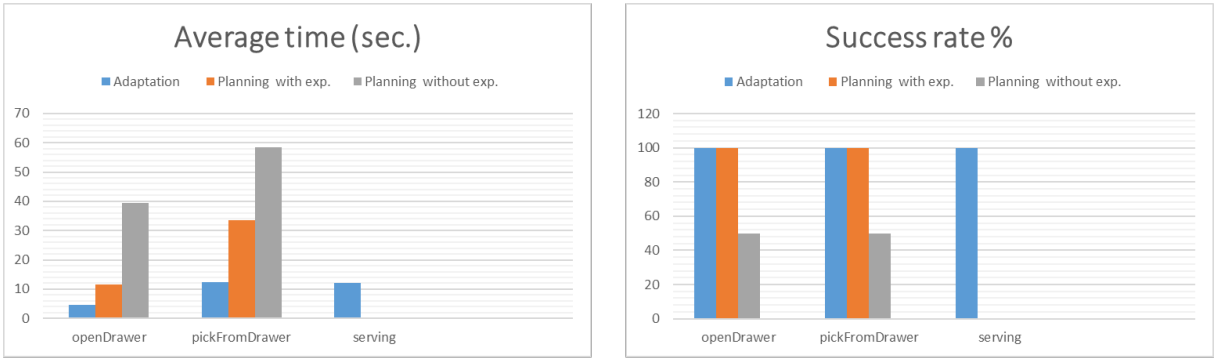


Figure 5.13: (a) Average time for *openDrawer, pickUp from the drawer* and *serving* skills using adaptation and planning with and without experiential knowledge. (b) Success rate for the each skill.

5.8.3 Results

The perception, planning, and adaptation modules are used with the knowledge for guidance, including experiential knowledge, to successfully execute the proposed tasks. As illustrated in Fig. 5.13 and Table 5.1, some factors have been considered to compare between adaptation and planning with and without guidance from the experiential knowledge, such as time and success rate. Three main skills are used for this evaluation, *openDrawer*, *pickUp* from the drawer and *serving* the *can* to a customer in the tray.

The adaptation of a planned-based trajectory is used to open the second drawer in the file cabinet in the storage task meanwhile a human demonstration is adapted to pour the can contents in the serving task. For the proposed skills, the number of calls to the collision check module to verify the feasibility of the path toward the goal configuration for the adaptation is always less than the number of calls required for the motion planning. The reason behind that is the exploration process done while planning the trajectory of an action using sampling-based method, i.e. a huge number of configurations need to be evaluated in order to find a collision-free path. For skills composed of several actions (each one with its own trajectory) this is even more relevant.

We executed each skill 10 times. The similarity check reasoning process based on belief states of the world takes around 2.5 seconds over the PMK framework. That means, the total time of adaptation, including similarity check reasoning, is still less than the planning time. The reasoning process was run on an Intel Core i7-4500U 2.40 GHz CPU with 8 GB memory.

Fig. 5.13-b shows that the adaptation and planning with experiential knowledge have a high percentage of success rate, while this is not the case of planning without it. The reason behind

that is the exploration process required, i.e. when no information is given, there is a need to explore all the possibilities regarding potential grasps (here two type of grasps were considered, top and side) until a solution is found.

5.9 Discussion

5.9.1 Discussion about the results

The SkillMaN framework provides services that are necessary for every-day activity tasks, with a knowledge source, reasoning engine and skills descriptions. It has been tested with a TIAGo mobile manipulator in a lab but it is flexible enough to be extended to other robots (with different kinematics and control architecture) or other environments (even though, the generation of semantic descriptions of the environments requires of manual processing).

The strategy of the task manager allows the the robot to build the experience automatically. That means, initially, the planner can be used to build the robot experience. To set the initial scene to the planner, the initial state extractor, i.e., perception modules, localizes all the objects in the robot working space with respect to the world frame. Also, it is used to verify the satisfaction of some of the skills preconditions. For example, as shown in Fig. 5.7, the precondition to apply the *openDrawer* skill in a drawer is that the drawer must be closed. This dynamic information is retrieved from the RFID memory. The localization of the objects is done using the integration between RFID and camera. By describing the robot and working space, the geometric modules, i.e. inverse kinematic, motion planning, collision check and object placement, in assistant system can be called in a structured way from the task manager to generate a collision-free path of each symbolic skill. Then, store the trajectory in the DB. The outcome of perception and semantic inference are used to reason about the features of the objects (e.g, the handle type of the drawer), the robot (e.g, the proper location to avoid collision with the target object), the execution (i.e., the tips of how to execute the task). Then, the DMP adapts the trajectory.

Further, to plan the storage task shown in scenario one, the robot needs to call the general task planning to compute the sequence of skills, which requires to run a general perception system (RFID) to set the initial scene for the planner. Then query over the knowledge module to illustrate the task constraints such as the top-grasp is used to pickUp/putDown the can from/to the drawer in the file cabinet. Moreover, query over the geometric module to reason on the feasibility of each skill and generating its path. In the case the robot has experience of how to execute such skill in a certain situation (similarity check), the adaptation module is used to adapt/imitate the skills.

As illustrated in Fig. 5.13, the planning process can be used to explore (without experience) how to grasp the cans. That means, the robot can start with a side-grasp or top-grasp with

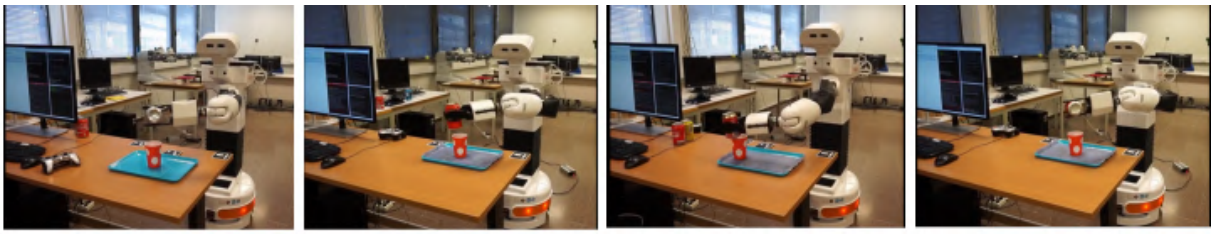


Figure 5.14: Adaptation process for serving skill with different poses.

different parameters (angles). We assume the robot changes the grasp type if a failure occurs (i.e., if the side-grasp return false, the top-grasp is selected). For *pickUp from the drawer* skill, the successful rate is half because of selecting a side-grasp that collides with the drawer body and it reports fail. Then the top-grasp is used and it reports success.

Several final positions for serving task are proposed to test the flexibility of the adaptation approach. It successfully presents the contents of the can inside the cup, as shown in Fig. 5.14.

As demonstrated in Fig. 5.13, and based on experimental evaluation shown in the Table 5.1, the use of the adaptation method is powerful for every-day tasks when a trajectory already exists in the DB. It is demonstrated in the calling of the collision check module for feasibility verification and execution time.

5.9.2 Challenges and limitations

The SkillMaN framework has been able to cope with the following challenges in the scenarios studied:

1. *Detection of NLOS:* The objects are not always in the robot sight which requires a sensor like RFID to be used to figure out the hidden objects such as in the case of the is proposed for a serving task scenario.

2. *Check skill feasibility:* The skills are not always feasible which requires a geometric reasoning with guidance from knowledge to check the availability to apply them in such situations. For instance, some geometric challenges have been considered in the scenarios:

   (a) In the storage task, a challenge of how to pick the can and place it inside a drawer This requires, based on geometric-skills experience, a top-grasp.

   (b) In the serving task, a challenge of how to serve the can to pour its contents in the cup. This requires changing the type of grasp from top to side. Moreover, a free room to place the can to change the grasping type.

These challenges show the flexibility of SkillMaN to work in a semi-unstructured environments as presented in the proposed scenarios or a fully unstructured environments.

The SkillMaN has a limitation regarding motion adaptation. Although the proposed primitives are quite general and can be used in several scenarios in manipulation domain, it has some limitations regarding the flexibility to be used in complex manipulation scenarios which includes special parameterization, like *putOnTop* skill that requires to consider some parameters such as height, distance and spatial relations which includes two objects on top of each other.

5.9.3 System flexibility

The proposed system is flexible enough to e.g.:

1. use untagged-based perception systems, such as the one presented in (Konidaris et al., 2018),
2. extend the way of describing the skills, besides describing using gestures or as actions in a planning system,
3. add more skills, like grasp an object in a cluttered environment allowing the interaction with the obstacles as presented in (Muhayyudin et al., 2018), and
4. to adapt more skills (e.g. in our system, for instance, the robot adapts the *openDrawer* skill based on the status of the drawer to either open or close it).

Regarding the proposed perception system, it is enhanced by the use of RFID technology with tags that include a physical storage medium and several antennas to cover different regions of the lab. The use of this technology increases the richness of perception capabilities in several aspects, such as:

1. being able to know the region where an object is located (without knowing its exact pose, just by using the antennas signals) which enhances the planning capabilities by allowing the robot to start planning and executing the task without a complete knowledge of the scene;
2. being able to perceive objects without line of sight (like in the serve scenario where the robot knew a full can was stored in the second closed drawer).

5.9.4 Using recovery module within SkillMaN

To tackle failure problems, failure detection and recovery ontologies presented in chapter four have been integrated within the SkillMaN framework to let the robot interpret such failures that

can occur and repair them. For this purpose, another motivating example has been introduced to justify this approach using a contingent-based planning approach. It is a continuation of our previous work presented in 4, and involves an autonomous mobile manipulator performing manipulation tasks in a cluttered environment that may contain challenges for the robot. The integration of our proposed knowledge-based reasoning mechanism and contingent-based planning approach has been implemented for TIAGo robot and demonstrated on some test scenarios.

This example is a step towards a more automated, reasoning and knowledge-driven approach to failure handling. The reasoning is integrated into the perception-action loop of the robot, and able to guide a plan repair process to resume or repeat a task after a failure. Further details are:

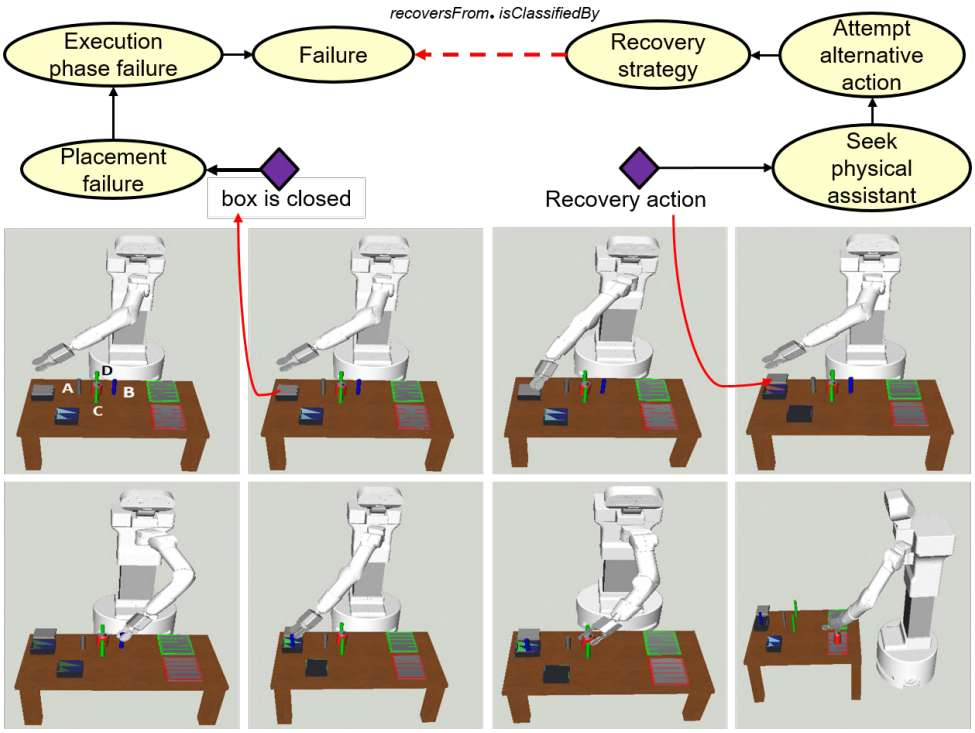
- *State interpretation based on observation:* Integration of perception and the knowledge modeling of failures. We show how perception can flag abnormal events and describe their nature in terms of previously defined conceptualizations of failure.
- *Plan repair guided by reasoning:* Failure descriptions are the input of reasoning processes that map failure types to possible recovery strategies via classification reasoning. The conceptualization of recovery strategies present in the failure ontology allows posing repair and replanning queries to a contingency task planner.
- *Communication and requesting assistance:* One of the recovery strategies we have included in the ontology is delegating a difficult task to another agent, and in particular asking a human observer for help. A reasoning based approach that accounts for the robot's capabilities allows deciding when such a strategy is necessary and what should be communicated to the human from whom assistance is requested.

A) Motivation example description

An example presented in our previous work (Akbari et al., 2020) is considered here to illustrate how the reasoning for failure interpretation is done in conditional plan. The task is to store an object (a cylinder labeled as A) in a given tray according to its color. Cylinder A may be among other objects, which have to be manipulated according to its nature.

The scene depicted in Fig. 5.15 shows the initial belief state of the mobile manipulation problem, where cylinder A is painted in gray because its color is uncertain (it could be red or green), it is not known whether the can is filled or not (and hence whether the can should be pushed or picked instead), nor if the containers are open or closed (and hence if the placement of an object inside the box can be directly performed or not). The goal to transfer the cylinder A to either the red or the green tray according to its color, will require of a plan that involves also the manipulation of some of the other objects. Some particular placements regions are allocated for the manipulatable objects if required:

Figure 5.15: The sequence of snapshots from planning process. The first image shows the manipulation example where the goal is to transfer the gray cylinder (labeled as A) to one of the trays w.r.t its color.



- The green cylinders must be placed on the green tray,
- The red can must be placed over the red tray,
- The blue cylinder must be placed within the containers.

B) Skill description

The following skills types are considered: manipulation skills, sensing actions and reasoning actions.

Manipulation skills require motion, and can be done by either the robot or a person. The following manipulation skills are considered in this example:

- *Transit:* a skill done by the robot to travel from one configuration to another without an attached object.

Figure 5.16: The conditional plan results from the planning process. a) flowchart describing the plan obtained by the contingent FF. b) flowchart added when executing the plan for monitoring and repair if necessary the action outcomes shown in red in the plan.

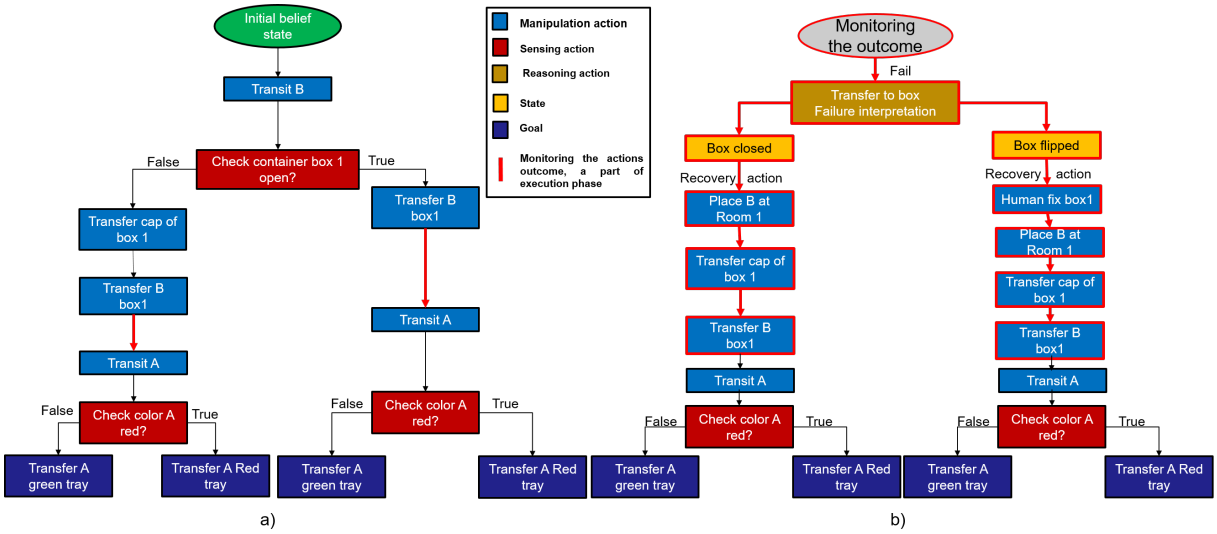


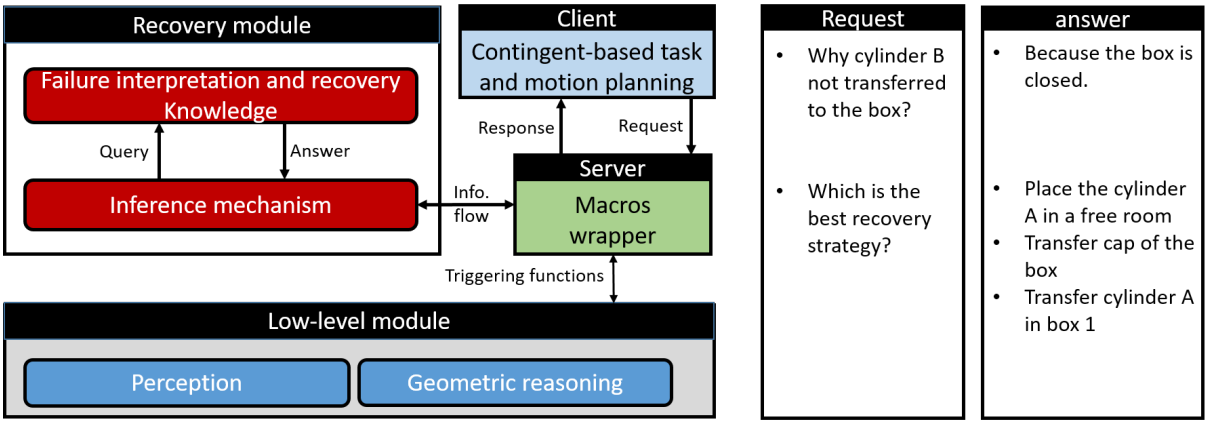
Figure 5.17: The executable plan.

- *Transfer:* a skill done by the robot to move an attached object between poses.
- *Push:* a skill done by the robot to push an object from one pose to another one.
- *Open:* a skill done by the robot to open a box-like container (articulated cap with prismatic joint is assumed with two positions corresponding to fully closed and fully opened, the state being stored in the containers objects features).
- *HumanTransfer:* a skill done by a person to transfer/push an object to the robot workspace.
- *HumanOpen:* a skill done by a person to open a box-like container.

Sensing actions do not involve motion, and are devoted to observe object status. The observation is done at run-time. The sensing actions considered in the example are :

- *CheckColor:* a sensing action done by a robot to determine object color.
- *CheckPose:* a sensing action done by a robot to determine object pose.

Figure 5.18: The integration of contingency plan with recovery knowledge.



- *CheckContainer:* a sensing action done by a person to evaluate whether a container is open or not.
- *CheckCan:* a sensing action done by a person to evaluate whether can-like objects are filled or not.

The reasoning actions are devoted to interpret execution phase failures, providing a strategy (or maybe more according to the failure type) for recovery. The reasoning process is done at run-time. In this example the following reasoning action is considered:

- *Transfer-to-box-FailureInterpretation:* a reasoning action over failure ontology interprets failure cause while transferring an object in a container, and provides a recovery action for the robot or human depending on the interpretation process.

The contingent Fast-Forward planner finds a the conditional tree of manipulation skill plans to execute the task, sensing actions will be involved in the branching nodes, and reasoning actions will be associated to the monitoring of the execution of the more sensitive manipulation skills.

C) The integration of recovery module in SkillMaN with conditional plan

The complete conditional tree of plans is represented in Fig. 5.16. While the planning process is taking place, there are several challenges in terms of interpretation of the actual state which

is captured by a sensory action and handled by the proposed state interpretation. The main challenge is to include in the plan the knowledge about who is responsible for the recovery action execution (i.e., human or robot). That means, for instance, that the robot can autonomously repair the failure of transferring the blue cylinder B inside the box if it interprets that the box is closed.

The plan automatically obtains branching nodes in which sensing and/or reasoning actions are assigned to monitor manipulation skill outcomes in a semi-automatic way. Logically, the sensing/reasoning actions should be assigned after each manipulation skill, however this increases the computational cost. Instead, we assign them to some manipulation skill that are expected to have a high probability of failure.

The reasoning action *Transfer-to-box-FailureInterpretation* is assigned to the action *Transfer-B-box1* based on monitoring the result of the skill execution. The result has a Boolean outcome, the success sequence is automatically obtained by the contingent FF planner as shown in Fig. 5.16-a). If, while monitoring, a failure occurs, the reasoning action interprets the cause of the failure which is a failed task execution that has unmanifested postconditions (i.e., *NonrealizedSituation*). The recovery strategies have been provided based on the current status of the box (closed or flipped) as shown in Fig. 5.16-b). Recovery strategies could be either to ask help from humans or the robot recovers the failure by itself. The other FALSE outcome branch of the sensing action *CheckContainerBox1* states the box is already closed and based on the planner, the transfer of the box cap is applied. The final executable plan generated by the planning system has a set of feasible manipulation skills as shown in Fig. 5.17.

The planner may ask the ontology questions about *why cylinder B has not been transferred to the box?*, once it interprets the current situation (i.e., the box is closed), then the next request is *which is the best recovery strategy?*. These requests of the planner are handled and answers by retrieving information, updating/deleting or reasoning over it. As shown in Fig. 5.18-a, the request-answer relation is done using the service-client communication of ROS (Robot Operating System, www.ros.org), and in Fig. 5.18-b, the request-answer queries of the planner are described.

5.10 Summary of the chapter

This framework discusses the importance to integrate perception, planning, knowledge-based reasoning (including experience), in a skill-based manipulation framework to let the robot automatically perform the tasks that include every-day activities. Moreover, the framework also includes the procedures to determine how to manage the data required to efficiently perform the tasks. Two examples have been introduced. In both examples, a set of skills such as *pickUp, putDown, openDrawer* and *serving* are introduced. The first example with two scenarios including manipulation in indoor environment has been introduced to show the capabilities of the robot to use the proposed modules to execute the every-day tasks in semi-structured

environments. For every-day tasks, the adaptation method is powerful in terms of time when the robot already has experience of how to execute the task. For planning, experiential knowledge is used as a geometric-skill experience to facilitate the planning process and reduce the cost that increases due to the exploration process. In the second example, the integration of the framework with a recovery module has been done, besides the manipulation skills, a sensing and reasoning actions are introduced to interact with the physical environment and monitor the results of the manipulation skills. The system shows flexibility to be adapted in several environments and robotic structures.

5.11 Enhancement

Future work will be how to increase the adaptation capability to work with more complex situations which include spatial relations. Also, finding out the way to automate the process of establishing a new concept of an unknown environmental entity.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The present thesis has developed several frameworks based on perception, reasoning, learning, and planning to address the increasing challenges of robotic manipulation problems. Different sorts of modeling and reasoning processes have been also proposed inside the frameworks to come up with a feasible manipulation plan. To sum up, the challenges of robotic manipulation problems considered in the thesis are summarized as follows:

- **Table-top manipulation problems**: The need to reason on the current state of the world in order to apply one action or another (like pick or push an object to take it apart) in a table-top manipulation scenario. KTMP framework offers the way to encode the knowledge and reason upon it.
- **Assembly manipulation problems**: The need to reason on the result of actions or the need to evaluate actions preconditions (like the feasibility of a given grasp to execute a pick action) in an assembly task. FailRecOnt framework offers a flexible reasoning tool that is perfectly adapted to knowledge-driven planning schemes.
- **Mobile-based every-day manipulation problems**: The need to make the robot aware of situation similarities to efficiently re-use previous known ways to execute actions by adapting the robot motions to stored patterns (like opening one drawer once the robot knows how to open another). The SkillMaN framework integrates the previous tools with a similarity situation evaluator and a motion adaptation tool.

Two approaches of planning are used in this thesis, heuristic-based approaches (i.e., Fast Forward (FF) and contingent-based FF), and knowledge-based planner provided by the KowRob

group. The execution of manipulation tasks with knowledge-based planning approaches not explicitly prepared for TAMP, like (Tenorth and Beetz, 2009), can be a challenge because this would require, on the one hand, from the knowledge perspective, to provide all the components in a way that they match with their planning system. On the other hand, from the planning perspective, they would require the definition of the recipes (strategies) for executing the tasks (sequence of actions), including all possible strategies of execution and the way to switch between them when required, which can be a very expensive process, especially for tasks that need long sequences of actions, such as those involving manipulation in cluttered environments. Some other planning approaches rely on PDDL and on planning strategies best fit to cope with difficult task planning challenges, like those found in the manipulation domains, although the use of PDDL implies a closed-world assumption, which precludes their use in more dynamic environments that could require perception and knowledge-based geometric reasoning. PMK allows to break the closed world assumption of classical-based manipulation planning approaches.

Robots, like any other agent, sometimes fail. Knowledge-based robots can recover from failure by reasoning whether to try once more, to try something else or to move to other tasks. We argued that the choice has to be based on the conceptual (ontology), the planning (task) and the execution (feasibility) levels. This requires to integrate traditional robotics domains (the robot has to act) and AI (the robot has to plan) with unplanned situations (the robot is in an unexpected state). The integration of all these views raises a variety of research questions, and so does our work which addresses only part of this research topic. For instance, hardware related failures that may appear when e.g. the robot in a real environment requires to be re-calibrated (gripper or arm). Also, software agent related failures, that may appear when e.g. the robot has software components that fail like when an algorithm is not able to extract the proper features. One of the focus of this thesis has been the justification and development of FailRecOnt, a general and reliable framework for failure and recovery management.

Foundationally, the proposed ontologies are modeled under SUMO and DUL foundations. We found that SUMO concepts have some limitations in the terms descriptions and some missing vocabulary that we proposed in PMK ontology. On the contrary, DUL has a well-structure and wide range of meaningful concepts that can be flexibly used in such domains.

To increase the robot autonomy, the integration of services that are necessary for every-day activity tasks, with a knowledge source, reasoning engine and skills descriptions are very important. Moreover, the use of experiential knowledge is very useful when the robot encounters the same situation. Moreover, the adaptation methodology saves time comparing to motion planning. However, huge efforts must be done in this line to enhance the capabilities of learning from demonstrations in such situations.

Finally, the frameworks have been illustrated to show the main tools and the flow of information among modules used to perception, reasoning, learning and planning levels. Concerning the robotic systems, we have tested our results with the robots *Yumi ABB* and *TIAGo PAL*. All the related results are shown in videos in URL: https://www.youtube.com/channel/UC6lZ7d7qm5wh5v3fsbEIFgg?view_as=subscriber and alternatively in https://sir.upc.

edu/projects/kautham/Videos.html.

6.2 Future Work

Along with the conclusion points stated above, the current thesis, moreover, opens new research problems that require further consideration such as:

From the many issues that the KTMP framework raises, in the future we aim to

- increase the reasoning capabilities of the framework;
- reduce the uncertainty of the low-level information by using deep learning techniques.
- increase the abstract concepts to include some notions like *behavior* which is important to be used in constraint-based manipulation planning.
- benchmark PMK with other approaches that includes more concrete metrics in relation to overall system performance, compatibility with other frameworks (e.g. planners, perception systems), extensibility, reusability, scalability, types of applications it can be applied to.

From the many issues that the FailRecOnt framework raises, in the future we aim to

- enrich the causal explanatory module;
- improve the search for an optimal match between what is known about a detected failure and the recovery strategies;
- include recovery strategies from a false belief state (e.g, caused by false detection), and
- optimize the interconnections among the FailRecOnt submodules.

From the many issues that the SkillMaN framework raises, in the future we aim to

- increase the robot autonomy by implementing a library of actions/skills and a sophisticated reasoning mechanism to allow the robot to reason on the best action/skill that can be used in the current situation;
- increase the adaptation capabilities to cover more complex manipulation problems which requires spatial reasoning;

- build a failure-based experiential knowledge that allows the robot to prevent the repetition of its mistakes.

Some of these works are already in preparation. Their development shall make the robots smarter and more adaptive.

Appendices

Appendix A

Vocabulary of PMK levels

This appendix defines the formal vocabularies of the PMK levels. Some vocabularies are taken literally from the standard (IEEE-SA, 2015; Olszewska et al., 2017), and others, not covered by the standard and required for the manipulation domain, are proposed here.

- *Quantity:* (IEEE-SA, 2015) Any specification of how many or how much of something there is. Accordingly, there are two subclasses of quantity: number (how many) and physical quantity (how much).
- *Quantity Aggregation:* A single quantity that represents a set of quantities such as Pose that represents the 3D location of the objects in the environment.
- *Attribute:* (IEEE-SA, 2015) Abstract qualities that can not or are chosen not to be considered as sub classes of Object.
- *Artifact Component:* (IEEE-SA, 2015) Representation of the parts of the workspace object in the world.
- *Artifact:* (IEEE-SA, 2015) An object that is the product of a making.
- *Collection:* (IEEE-SA, 2015) Collections have members like classes, but, unlike classes, they have a position in space-time and members can be added and subtracted without thereby changing the identity of the collection.
- *Robot Component:* Representation of the parts of the robot in the world.
- *Robot:* (IEEE-SA, 2015) A device in the world that is responsible for executing the tasks.
- *Robot Group:* (IEEE-SA, 2015) A group of robots organized to achieve at least one common goal.

- *Measuring Device Component:* Representation of the parts of the measuring device (sensor).
- *Measuring Device:* (IEEE-SA, 2015) Any device whose purpose is to measure a physical quantity.
- *Device Group:* A group of measuring devices that supply the robot information to achieve one common goal.
- *Region:* (IEEE-SA, 2015) A topographic location. Regions encompass surfaces of objects, imaginary places, and geographic areas.
- *Physical Environment:* (IEEE-SA, 2015) A physical environment is an object that has at least one specific part: a region in which it is located. In addition, a physical environment relates to at least one reference object based on which region is defined.
- *Semantic Environment:* A physical environment with data (feature) of the artifacts.
- *Spatial Context:* The circumstances that form the setting for an event that is related to space and which it can be fully understood. Specifically, a representation of the world in terms of space.
- *Temporal Context:* The circumstances that form the setting for an event that is related to time and which it can be fully understood. Specifically, a representation of the world in terms of time.
- *Situation:* The physical object situation in the environment that represents spatially and temporally the relation of the objects each others.
- *Atomic Function:* A representation of the processes for motion, manipulation and perception, such as task planners, motion planners or perceptual algorithms. Moreover, it includes primitive actions, preconditions and postconditions related to task planning.
- *Sub-task* (Olszewska et al., 2017) The summarization of the typical definition is, a representation of a short-term sequence of action.
- *Task* (Olszewska et al., 2017) The summarization of the typical definition is, a representation of a long-term goals of the symbolic preconditions and effects.

Appendix B

Description Logic

B.1 An introduction to ontologies

An *ontology* is a description of a conceptualization using a logical formalism. That is, an ontology defines categories – also known as concepts – of entities relevant for some domain of discourse, and properties of and relations between these entities, via logical axioms and thus enables reasoning with these concepts and entities.

An important aspect of an ontological description of a domain is a distinction between what can be called data, versus what can be called knowledge. For example, to assert that Bob has a particular birthdate, or that Bob is a human, is a matter of data. To assert that anything which is human must also have a birthdate is a matter of knowledge. In this context here, data refers to contingent entities and relations; it is not necessary for Bob to exist. However, if he does, then it is necessary that he has a birthdate – this comes from knowledge of a general pattern, that is part of the definition of the concept `Human`. This distinction also highlights a difference between an ontology and other knowledge representations such as databases or knowledge graphs: the ontology is not about simply containing data, or relations between a set of individuals.

The data/knowledge distinction is sometimes referred to as such, but often in the literature, and in this thesis as well, the terminology of ABox and TBox is used. An *ABox* (*assertion box*) corresponds to what was previously described as data: it contains *axioms* about what individuals exist, their relations, and their properties. The *TBox* (*terminology box*) corresponds to knowledge of what concepts exist, how these concepts are defined, and what kind of relations and properties exist.

Typical ontological reasoning tasks include *satisfiability*, *subsumption*, and *classification*. *Satisfiability* means checking whether a set of axioms has a model, that is, whether there can

exist some network of individuals and relations between them such that the axioms are satisfied. *Subsumtion* reasoning answers queries of the form *given that an individual belongs to concept* `A`*, is it also guaranteed to belong to concept* `B`? Both satisfiability and subsumption are examples of TBox reasoning (reasoning about concepts). *Classification* is an example of ABox reasoning as it answers queries of the form *given some data about individual X and an axiomatic description of concept* `C`*, is it certain that X belongs to* `C`?

Any logical language can be used to write an ontology, including full first order logic. However, many existing ontologies use a formalism called *Description Logics* (DL), and in particular fragments of such logics defined by the *Ontology Web Language* (OWL). Unlike first order logic, these fragments are decidable – so any reasoning queries are guaranteed to terminate – and efficient in practice reasoners exist for them. Our ontological modelling in this thesis also makes use of a DL fragment, specifically OWL2.

Note that relations between entities are referred to as *object properties*, and sometimes *roles*, in OWL; we use the same terminology in this thesis. So for example, we would say that the object property *hasPart* links a `Car` individual to a `Motor` individual. Note that object properties are directed, from one individual to another.

B.2 Description Logic notion

Table B.1: DL notation

Notation	*Type*	*Interpretation*
	Class constant	
$\top$	*top*	contains all individuals, often referred to as *Thing*, or *Entity*
$\bot$	*bottom*	the *empty concept*, it contains no individuals
	Propositional class	
$C \sqcap D$	*intersection*	contains all individuals that are instances of class `C` and `D`
$C \sqcup D$	*union*	contains all individuals that are instances of either class `C` or `D`
$\neg C$	*complement*	contains all individuals that are not instances of class `C`
	Restriction class	
$\exists r.C$	*existential quantification*	contains all individuals that are linked by property *r* to an individual that is an instance of `C`

Table B.1 Continued: DL notation

Notation	*Type*	*Interpretation*
$\forall r.C$	*universal quantification*	contains all individuals that are linked by property *r* only to individuals that are instances of C
$r : X$	*value quantification*	contains all individuals that are linked by property *r* to individual *X*
$\geq nr.C$	*minimum cardinality*	contains all individuals that are connected by property *p* to at least n different individuals that are instances of class C
$\leq nr.C$	*maximum cardinality*	contains all individuals that are connected by property *p* to at most n different individuals that are instances of class C
$= nr.C$	*exact cardinality*	contains all individuals that are connected by property *p* to exactly n different individuals that are instances of class C
	Class axiom	
$C \equiv D$	*equivalence*	class C and class D are semantically equivalent, i.e., each instance of one of the classes is also an instance of the other class
$C \sqsubseteq D$	*subclass*	class C is more specific then class D, i.e., each instance of class C is also an instance of class D
	Property axiom	
$r \sqsubseteq s$	*subproperty*	if an individual *X* is linked by *r* to an individual *Y*, then *X* is also linked by *s* to *Y*
$r_0 \circ r_1 \sqsubseteq s$	*role-chain*	if an individual *X* is linked by a sequence of properties r_0, r_1 with an individual *Y*, then *X* is also linked with *Y* by the property *s*
	Assertion	
$C(X)$	*class assertion*	individual *X* is an instance of class C
$r(X, Y)$ or XrY	*property assertion*	individual *X* is linked by a property *r* to individual *Y*

Bibliography

Akbari, A., Diab, M., and Rosell, J. (2020). Contingent task and motion planning under uncertainty for human–robot interactions. *Applied Sciences*, 10(5):1665.

Akbari, A., Lagriffoul, F., and Rosell, J. (2018a). Combined heuristic task and motion planning for bi-manual robots. *Autonomous Robots*, 32(9-10):1–16.

Akbari, A., Muhayyuddin, and Rosell, J. (2016a). Reasoning-based evaluation of manipulation actions for efficient task planning. In *Robot 2015: Second Iberian Robotics Conference*, pages 69–80. Springer.

Akbari, A., Muhayyuddin, and Rosell, J. (2016b). Task planning using physics-based heuristics on manipulation actions. In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, pages 1–8. IEEE.

Akbari, A., Muhayyuddin, and Rosell, J. (2018b). Knowledge-oriented task and motion planning for multiple mobile robots. *Journal of Experimental & Theoretical Artificial Intelligence*, 0(0):1–26.

Akbari, A., Muhayyudin, and Rosell, J. (2016c). Task planning using physics-based heuristics on manipulation actions. In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, pages 1–8. IEEE.

Akbari, A., Rosell, J., et al. (2018c). κ-pmp: Enhancing physics-based motion planners with knowledge-based reasoning. *Journal of Intelligent & Robotic Systems*, 91(3-4):459–477.

Akbari, A., L. F. R. (2018). Combined heuristic task and motion planning for bi-manual robots. *Autonomous robots*, pages 1–16.

Alami, R., Gharbi, M., Vadant, B., Lallement, R., and Suarez, A. (2014). On human-aware task and motion planning abilities for a teammate robot. In *Human-Robot Collaboration for Industrial Manufacturing Workshop, RSS 2014*.

Alili, S., Pandey, A. K., Sisbot, E. A., and Alami, R. (2010). Interleaving symbolic and geometric reasoning for a robotic assistant. In *ICAPS Workshop on Combining Action and Motion Planning*, volume 3, pages 4–3. Citeseer.

Alirezaie, M. (2015). *Bridging the Semantic Gap between Sensor Data and Ontological Knowledge*. PhD thesis, Örebro university.

Anagnostopoulos, C. and Hadjiefthymiades, S. (2009). Advanced inference in situation-aware computing. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(5):1108–1115.

Antoniou, G. and van Harmelen, F. (2004). *Web Ontology Language: OWL*, pages 67–92. Springer Berlin Heidelberg, Berlin, Heidelberg.

Azizi, V., Kimmel, A., Bekris, K., and Kapadia, M. (2017). Geometric reachability analysis for grasp planning in cluttered scenes for varying end-effectors. In *Automation Science and Engineering (CASE), 2017 13th IEEE Conference on*, pages 764–769. IEEE.

Baader, F., Horrocks, I., Lutz, C., and Sattler, U. (2017). *An Introduction to Description Logic*. Cambridge University Press, United Kingdom.

Balakirsky, S., Kootbally, Z., Kramer, T., Pietromartire, A., Schlenoff, C., and Gupta, S. (2013). Knowledge driven robotics for kitting applications. *Robotics and Autonomous Systems*, 61(11):1205–1214.

Beetz, M., Bálint-Benczédi, F., Blodow, N., Nyga, D., Wiedemeyer, T., and Marton, Z.-C. (2015a). Robosherlock: Unstructured information processing for robot perception. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1549–1556. IEEE.

Beetz, M., Jain, D., Mosenlechner, L., Tenorth, M., Kunze, L., Blodow, N., and Pangercic, D. (2012). Cognition-enabled autonomous robot control for the realization of home chore task intelligence. *Proceedings of the IEEE*, 100(8):2454–2471.

Beetz, M., Tenorth, M., and Winkler, J. (2015b). Open-ease. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1983–1990. IEEE.

Beßler, D., Pomarlan, M., and Beetz, M. (2018a). Owl-enabled assembly planning for robotic agents. In *Proceedings of the 2018 International Conference on Autonomous Agents*, AAMAS '18.

Beßler, D., Pomarlan, M., and Beetz, M. (2018b). Owl-enabled assembly planning for robotic agents. In *Proceedings of the 2018 International Conference on Autonomous Agents*, AAMAS '18, Stockholm, Sweden. Finalist for the Best Robotics Paper Award.

Beßler, D., Pomarlan, M., and Beetz, M. (2018c). Owl-enabled assembly planning for robotic agents. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1684–1692. International Foundation for Autonomous Agents and Multiagent Systems.

Beßler, D., Porzel, R., Mihai, P., and Beetz, M. (2019). Foundational models for manipulation activity parsing.

Blum, A. L. and Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial intelligence*, 90(1):281–300.

Brafman, R. and Hoffmann, J. (2004). Conformant planning via heuristic forward search: A new approach. In Koenig et al. (2004), pages 355–364.

Bruno, B., Chong, N. Y., Kamide, H., Kanoria, S., Lee, J., Lim, Y., Pandey, A. K., Papadopoulos, C., Papadopoulos, I., Pecora, F., et al. (2017). The caresses eu-japan project: making assistive robots culturally competent. In *Italian Forum of Ambient Assisted Living*, pages 151–169. Springer.

Bruno, B., Menicatti, R., Recchiuto, C. T., Lagrue, E., Pandey, A. K., and Sgorbissa, A. (2018). Culturally-competent human-robot verbal interaction. In *2018 15th International Conference on Ubiquitous Robots (UR)*, pages 388–395. IEEE.

Bruno, B., Recchiuto, C. T., Papadopoulos, I., Saffiotti, A., Koulouglioti, C., Menicatti, R., Mastrogiovanni, F., Zaccaria, R., and Sgorbissa, A. (2019). Knowledge representation for culturally competent personal robots: requirements, design principles, implementation, and assessment. *International Journal of Social Robotics*, pages 1–24.

Bryce, D., Kambhampati, S., and Smith, D. E. (2006). Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research*, 26:35–99.

Cambon, S., Alami, R., and Gravot, F. (2009a). A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126.

Cambon, S., Alami, R., and Gravot, F. (2009b). A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126.

Chang, C., Sheu, J., Chen, Y., and Chang, S. (2009). An obstacle-free and power-efficient deployment algorithm for wireless sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(4):795–806.

Chatterjee, R., Takao, I., Matsuno, F., and Tadokoro, S. (2005). Robot description ontology and bases for surface locomotion evaluation. In *IEEE International Safety, Security and Rescue Rototics, Workshop, 2005.*, pages 242–247.

Dantam, N., Kingston, Z. K., Chaudhuri, S., and Kavraki, L. E. (2016a). Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*.

Dantam, N. T., Kingston, Z. K., Chaudhuri, S., and Kavraki, L. E. (2016b). Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*, pages 1–6.

Dargie, W. (2009). Adaptive audio-based context recognition. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(4):715–725.

De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O. C. (2000). *Computational geometry*. Springer.

de Silva, L., Pandey, A. K., Gharbi, M., and Alami, R. (2013). Towards combining HTN planning and geometric task planning. In *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*.

Dearden, R. and Burbridge, C. (2014). Manipulation planning using learned symbolic state abstractions. *Robotics and Autonomous Systems*, 62(3):355–365.

Deyle, T. (2011). *Ultra High Frequency (UHF) Radio-Frequency Identification (RFID) for Robot Perception and Mobile Manipulation,*. PhD thesis, Georgia Institute of Technology.

Deyle, T., Reynolds, M. S., and Kemp, C. C. (2014). Finding and navigating to household objects with uhf rfid tags by optimizing rf signal strength. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2579–2586.

Diab, M., Akbari, A., Ud Din, M., and Rosell, J. (2019). PMK –A knowledge processing framework for autonomous robotics perception and manipulation. *Sensors*, 19(5):1166.

Diab, M., Muhayyuddin, Akbari, A., and Rosell, J. (Springer 2017). An ontology framework for physics-based manipulation planning. pages 452–464, Robot 2017: Second Iberian Robotics Conference.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dornhege, C., Eyerich, P., Keller, T., Trüg, S., Brenner, M., and Nebel, B. (2009). Semantic attachments for domain-independent planning systems. In *Proceedings of the Nineteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'09, pages 114–121. AAAI Press.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110.

E. Tosello, Z. Fan, A. G. C. E. P. (2018). Rtask: A cloudbased knwoledge engine for robot task and motion planning. Fields of Robotics, Journal, under review.

Elbanhawi, M. and Simic, M. (2014). A review: motion planning. *IEEE Transactions*, 2:56–77.

Englert, P. and Toussaint, M. (2018). Learning manipulation skills from a single demonstration. *The International Journal of Robotics Research*, 37(1):137–154.

Fiorini, S. R., Carbonera, J. L., GonÃğalves, P., Jorge, V. A., Rey, V. F., Haidegger, T., Abel, M., Redfield, S. A., Balakirsky, S., Ragavan, V., Li, H., Schlenoff, C., and Prestes, E. (2015). Extensions to the core ontology for robotics and automation. *Robotics and Computer-Integrated Manufacturing*.

Galindo, C., Fernandez-Madrigal, J.-A., Gonzalez, J., and Saffiotti, A. (2008). Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955 – 966. Semantic Knowledge in Robotics.

Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2015). FFRob: An efficient heuristic for task and motion planning. In *Algorithmic Foundations of Robotics XI*, pages 179–195. Springer.

Gemignani, G., Capobianco, R., Bastianelli, E., Bloisi, D. D., Iocchi, L., and Nardi, D. (2016). Living with robots: Interactive environmental knowledge acquisition. *Robotics and Autonomous Systems*, 78:1–16.

Gillani, M., Akbari, A., and Rosell, J. (2016). Physics-based motion planning: Evaluation criteria and benchmarking. In *Robot 2015: Second Iberian Robotics Conference*, pages 43–55. Springer.

Gonçalves, P. J. and Torres, P. M. (2015). Knowledge representation applied to robotic orthopedic surgery. *Robotics and Computer-Integrated Manufacturing*, 33:90–99.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5):907 – 928.

Guizzardi, G. (2005). Ontological foundations for structural conceptual models.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107.

Hauser, K. (2014). The minimum constraint removal problem with three robotics applications. *The International Journal of Robotics Research*, 33(1):5–17.

Hauser, K. and Latombe, J.-C. (2010). Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 29(7):897–915.

Hauser, K., Ng-Thow-Hing, V., and Gonzalez-Baños, H. (2010). Multi-modal motion planning for a humanoid robot manipulation task. In *Robotics Research*, pages 307–317. Springer.

He, K., Lahijanian, M., Kavraki, L. E., and Vardi, M. Y. (2015). Towards manipulation planning with temporal logic specifications. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 346–352. IEEE.

Heintz, F., Kvarnström, J., and Doherty, P. (2010). Bridging the sense-reasoning gap: Dyknow–stream-based middleware for knowledge processing. *Advanced Engineering Informatics*, 24(1):14–26.

Hertle, A. and Nebel, B. (2017). Identifying good poses when doing your household chores: Creation and exploitation of inverse surface reachability maps. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 6053–6058. IEEE.

Hoffmann, J. and Brafman, R. (2005). Contingent planning via heuristic forward search with implicit belief states. In *Proc. ICAPS*, volume 2005.

IEEE-SA (2015). IEEE Standard Ontologies for Robotics and Automation. pages 1–60. Special Issue on Artificial Intelligence and Robotics.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE.

Imran, M. and Young, B. (2015). The application of common logic based formal ontologies to assembly knowledge sharing. *Journal of intelligent manufacturing*, 26(1):139–158.

Jain, A. and Kemp, C. C. (2010). El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces. *Autonomous Robots*, 28(1):45.

Johnston, B., Yang, F., Mendoza, R., Chen, X., and Williams, M.-A. (2008a). Ontology based object categorization for robots. In Yamaguchi, T., editor, *Practical Aspects of Knowledge Management*, pages 219–231, Berlin, Heidelberg. Springer Berlin Heidelberg.

Johnston, B., Yang, F., Mendoza, R., Chen, X., and Williams, M.-A. (2008b). Ontology based object categorization for robots. In *International Conference on Practical Aspects of Knowledge Management*, pages 219–231. Springer.

Kaelbling, L. P. and Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *IEEE Int. Conf. on Robotics and Automation Robotics and Automation*, pages 1470–1477.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580.

Keleştemur, T., Yokoyama, N., Truong, J., Allaban, A. A., and Padir, T. (2019). System architecture for autonomous mobile manipulation of everyday objects in domestic environments. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 264–269.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98.

Koenig, S., Zilberstein, S., and Koehler, J., editors (2004). *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, Whistler, Canada.

Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289.

Kootbally, Z., Schlenoff, C., Lawler, C., Kramer, T., and Gupta, S. K. (2015). Towards robust assembly with knowledge representation for the planning domain definition language (pddl). *Robotics and Computer-Integrated Manufacturing*, 33:42–55.

Krieg-Brückner, B., Frese, U., Lüttich, K., Mandel, C., Mossakowski, T., and Ross, R. J. (2005). Specification of an ontology for route graphs. In Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., and Barkowsky, T., editors, *Spatial Cognition IV. Reasoning, Action, Interaction*, pages 390–412, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kuffner, J. J. and LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE.

Kunze, L. and Beetz, M. (2017). Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. *Artificial Intelligence*, 247:352 – 380. Special Issue on AI and Robotics.

Lagriffoul, F. and Andres, B. (2016). Combining task and motion planning: A culprit detection problem. *The International Journal of Robotics Research*, 35(8):890–927.

Lagriffoul, F., Dantam, N. T., Garrett, C., Akbari, A., Srivastava, S., and Kavraki, L. E. (2018). Platform-independent benchmarks for task and motion planning. *IEEE Robotics and Automation Letters*, 3(4):3765–3772.

Lagriffoul, F., Dimitrov, D., Bidot, J., Saffiotti, A., and Karlsson, L. (2014). Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 33(14):1726–1747.

Lagriffoul, F., Dimitrov, D., Saffiotti, A., and Karlsson, L. (2012). Constraint propagation on interval bounds for dealing with geometric backtracking. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 957–964. IEEE.

Lagriffoul, F., Karlsson, L., Bidot, J., and Saffiotti, R. (2013). Combining task and motion planning is not always a good idea. In *RSS Workshop on Combined Robot Motion Planning*.

LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400.

Lemaignan, S., Ros, R., Alami, R., and Beetz, M. (2011). What are you talking about? grounding dialogue in a perspective-aware robotic architecture. In *2011 RO-MAN*, pages 107–112. IEEE.

Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., and Beetz, M. (2010). Oro, a knowledge management platform for cognitive architectures in robotics. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553. IEEE.

Lemaignan, S., Ros, R., MÃűsenlechner, L., Alami, R., and Beetz, M. (2010). Oro, a knowledge management platform for cognitive architectures in robotics. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553.

Li, Y., Mac Namee, B., and Kelleher, J. (2010). Navigating the corridors of power: using rfid and compass sensors for robot localisation and navigation. *The 11th Towards Autonomous Robotic Systems (TAROS 2010)*.

Lim, G. H., Suh, I. H., and Suh, H. (2011). Ontology-based unified robot knowledge for service robots in indoor environments. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):492–509.

Lozano-Perez, T. (1983). Spatial planning: A configuration space approach. *IEEE transactions on computers*, C-32:108–120.

Maass, W., Behrendt, W., and Gangemi, A. (2007). Trading digital information goods based on semantic technologies. *Journal of Theoretical and Applied Electronic Commerce Research*, 2(3):18–35.

Maillot, N., Thonnat, M., and Boucher, A. (2004). Towards ontology-based cognitive vision. *Machine Vision and Applications*, 16(1):33–40.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). Wonderweb deliverable d18 ontology library. Technical report, IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web.

Menicatti, R., Bruno, B., and Sgorbissa, A. (2017). Modelling the influence of cultural information on vision-based human home activity recognition. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 32–38.

Migimatsu, T. and Bohg, J. (2020). Object-centric task and motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 5(2):844–851.

Muhayyudin, Moll, M., Kavraki, L., Rosell, J., et al. (2018). Randomized physics-based motion planning for grasping in cluttered and uncertain environments. *IEEE Robotics and Automation Letters*, 3(2):712–719.

Munawar, A., De Magistris, G., Pham, T.-H., Kimura, D., Tatsubori, M., Moriyama, T., Tachibana, R., and Booch, G. (2018). Maestrob: A robotics framework for integrated orchestration of low-level control and high-level reasoning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 527–534. IEEE.

Nam, C., Lee, S., Lee, J., Cheong, S. H., Kim, D. H., Kim, C., Kim, I., and Park, S. (2020). A software architecture for service robots manipulating objects in human environments. *IEEE Access*, 8:117900–117920.

Niles, I. and Pease, A. (2001a). Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9. ACM.

Niles, I. and Pease, A. (2001b). Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*, FOIS '01.

Olivares-Alarcos, A., Beßler, D., Khamis, A., Goncalves, P., Habib, M. K., Bermejo-Alonso, J., Barreto, M., Diab, M., Rosell, J., Quintas, J., et al. (2019). A review and comparison of ontology-based approaches to robot autonomy. *The Knowledge Engineering Review*, 34.

Olszewska, J. I., Barreto, M., Bermejo-Alonso, J., Carbonera, J., Chibani, A., Fiorini, S., Goncalves, P., Habib, M., Khamis, A., Olivares, A., de Freitas, E. P., Prestes, E., Ragavan, S. V., Redfield, S., Sanz, R., Spencer, B., and Li, H. (2017). Ontology for autonomous

robotics. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*.

Pangercic, D., Tenorth, M., Jain, D., and Beetz, M. (2010). Combining perception and knowledge processing for everyday manipulation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1065–1071.

Prestes, E., Carbonera, J. L., Fiorini, S. R., Jorge, V. A. M., Abel, M., Madhavan, R., Locoro, A., Goncalves, P., Barreto, M. E., Habib, M., Chibani, A., GÃľrard, S., Amirat, Y., and Schlenoff, C. (2013). Towards a core ontology for robotics and automation. *Robotics and Autonomous Systems*, 61(11):1193 – 1204. Ubiquitous Robotics.

Quispe, A. H., Amor, H. B., and Christensen, H. I. (2018). A taxonomy of benchmark tasks for robot manipulation. In *Robotics Research*, pages 405–421. Springer.

Rai, L. and Hong, J. (2013). Conceptual framework for knowledge-based decision migration in multi-component robots. *International Journal of Advanced Robotic Systems*, 10(5):237.

Rosell, J., Pérez, A., Aliakbar, A., Muhayyuddin, Palomo, L., and García, N. (2014). The Kautham Project: A teaching and research tool for robot motion planning. In *IEEE Int. Conf. on Emerging Technologies and Factory Automation*.

Ruiz-Sarmiento, J.-R., Galindo, C., and Gonzalez-Jimenez, J. (2017). Building multiversal semantic maps for mobile robot operation. *Knowledge-Based Systems*, 119:257–272.

Schlenoff, C., Prestes, E., Madhavan, R., Goncalves, P., Li, H., Balakirsky, S., Kramer, T., and Miguelanez, E. (2012). An IEEE standard ontology for robotics and automation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1337–1342. IEEE.

Sgorbissa, A., Papadopoulos, I., Bruno, B., Koulouglioti, C., and Recchiuto, C. (2018). Encoding guidelines for a culturally competent robot for elderly care. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1988–1995. IEEE.

Shah, R., Jiang, Y., Karnan, H., Briscoe-Martinez, G., Mulder, D., Gupta, R., Schlossman, R., Murphy, M., Hart, J. W., Sentis, L., et al. (2019). Solving service robot tasks: Ut austin villa@ home 2019 team report. *arXiv preprint arXiv:1909.06529*.

Shao, L., Migimatsu, T., Zhang, Q., Yang, K., and Bohg, J. (2020). Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*.

Siméon, T., Laumond, J.-P., Cortés, J., and Sahbani, A. (2004). Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):729–746.

Sisbot, E. A., Ros, R., and Alami, R. (2011). Situation assessment for human-robot interactive object manipulation. In *2011 RO-MAN*, pages 15–20. IEEE.

Srinivasa, S. S., Ferguson, D., Helfrich, C. J., Berenson, D., Collet, A., Diankov, R., Gallagher, G., Hollinger, G., Kuffner, J., and Weghe, M. V. (2010). Herb: a home exploring robotic butler. *Autonomous Robots*, 28(1):5.

Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., and Abbeel, P. (2014). Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE Int. Conf. on Robotics and Automation Robotics and Automation*, pages 639–646.

Stenmark, M., Haage, M., Topp, E. A., and Malec, J. (2018). Supporting semantic capture during kinesthetic teaching of collaborative industrial robots. *International Journal of Semantic Computing*, 12(01):167–186.

Stenmark, M., Malec, J., and Stolt, A. (2015). From high-level task descriptions to executable robot code. In *Intelligent Systems' 2014*, pages 189–202. Springer.

Stilman, M. and Kuffner, J. (2008). Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307.

Stilman, M., Schamburek, J.-U., Kuffner, J., and Asfour, T. (2007). Manipulation planning among movable obstacles. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3327–3332. IEEE.

Sucan, I., Moll, M., Kavraki, L. E., et al. (2012). The open motion planning library. *Robotics & Automation Magazine, IEEE*, 19(4):72–82.

Şucan, I. A. and Kavraki, L. E. (2009). Kinodynamic motion planning by interior-exterior cell exploration. In *Algorithmic Foundation of Robotics VIII*, pages 449–464. Springer.

Tenorth, M., Bartels, G., and Beetz, M. (2014). Knowledge-based specification of robot motions.

Tenorth, M. and Beetz, M. (2009). Knowrob: knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266.

Tenorth, M. and Beetz, M. (2017). Representations for robot knowledge in the Knowrob framework. *Artificial Intelligence*, 247:151 – 169. Special Issue on Artificial Intelligence and Robotics.

Topp, E. A. and Malec, J. (2018). A knowledge based approach to user support for robot programming. In *AI for Multimodal Human Robot Interaction Workshop within the Federated AI Meeting 2018 in Stockholm*, pages 31–34.

Warnier, M., Guitton, J., Lemaignan, S., and Alami, R. (2012). When the robot puts itself in your shoes. managing and exploiting human and robot beliefs. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 948–954. IEEE.

WIELEMAKER, J., SCHRIJVERS, T., TRISKA, M., and LAGER, T. (2012). SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96.

Wolfe, J., Marthi, B., and Russell, S. J. (2010). Combined task and motion planning for mobile manipulation. In *ICAPS*, pages 254–258.

Wu, Z. and Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yazdani, F., Kazhoyan, G., Bozcuoğlu, A. K., Haidu, A., Bálint-Benczédi, F., Beßler, D., Pomarlan, M., and Beetz, M. (2018). Cognition-enabled framework for mixed human-robot rescue teams. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1421–1428. IEEE.

Young Cheol Go and Joo-Chan Sohn (2005). Context modeling for intelligent robot services using rule and ontology. In *The 7th International Conference on Advanced Communication Technology, 2005, ICACT 2005.*, volume 2, pages 813–816.

Zaplana, I., Claret, J. A., and nez, L. B. (2018). Análisis cinemático de robots manipuladores redundantes: Aplicación a los robots kuka lwr 4+ y abb yumi. *Revista Iberoamericana de Automtíca e Informática industrial*, 15(2):192–202.