SkillMaN - A Skill-based Robotic Manipulation Framework based on Perception and Reasoning ⋆,⋆⋆

Mohammed Diab[a], Mihai Pomarlan[b], Daniel Beßler[b], Aliakbar Akbari[a], Jan Rosell[a], John Bateman[b] and Michael Beetz[b]

[a]*Institute of Industrial and Control Engineering, Universitat Politècnica de Catalunya, Barcelona.*
[b]*Universität Bremen, Bremen, Germany.*

ARTICLE INFO

Keywords:
Manipulation Planning,
Semantic Skill,
Navigation,
Every-day tasks,
Adaptation,
Knowledge-based Reasoning.

ABSTRACT

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. This paper proposes a planning and execution framework, called SkillMaN, for robotic manipulation tasks, 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.

1. Introduction

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 to collaborate with other operators. Working in an semi/unstructured environment means that the robot does not always have a model of the environment and 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 and open-drawer. Moreover, perception systems like vision, depth sensors, and others, can be used to model the geometry of objects in the environment and regularly update the status of dynamic parts. However, to figure out the entire world with perception before planning is quite computational expensive. Therefore, working with partial information becomes necessary, which requires a dominant perception capability to update key information as needed. Different types of sensors have their own limitations, so sensory integration methodologies are very helpful. Multi-sensory data integration aims to combine information from multiple sensory data or data derived from different sources. The goal of sensor integration is to obtain information which in some sense is better than the one obtained when the sources are used separately.

For every-day tasks, the experiential knowledge can play a significant role to make the robot capable to learn from its experience instead of repeatedly plan the same task with the same givens within the planning system, which could be a computational and time consuming process. An important research question raises: "*How can we adapt executions of these tasks to similar environments and agents?*" The answer is, a robot can imitate an observed action sequence or skill, like open a drawer, if it first understands the inherent characteristic features of each action. Such features need to reflect the semantics of the skill with a high degree of invariance between different demonstrations of the same skill. Then, a robot will be able to execute the skill in any appropriate similar situation.

1.1. Problem statement and proposal

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. 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 may become essential to facilitate the planning process in every-day tasks,

⋆This work was partially funded by the Spanish Government through the project DPI2016-80077-R and by Deutsche Forschungsgemeinschaft (DFG) through the Collaborative Research Center 1320, EASE. M. Diab is supported by the Spanish Government through the grants FPI 2017.

⋆Mohammed Diab, mohammed.diab@upc.edu
ORCID(s): 0000-0002-5743-5190 (M. Diab)

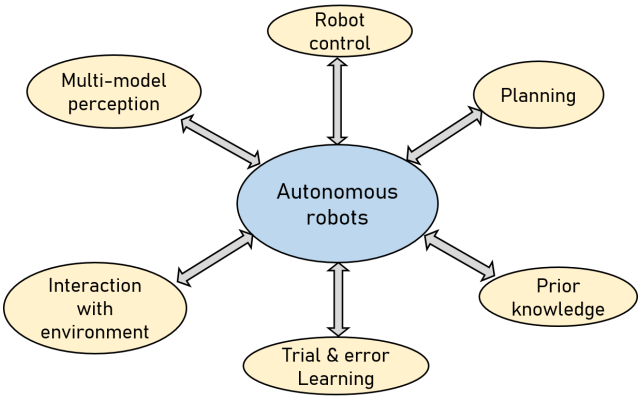


Figure 1: Autonomous robots requirements.

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 may 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
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.

1.2. Motivation examples

The mobile manipulator TIAGo is assumed to work in the semi-structured environment shown in Fig. 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.

2. Related work

Various studies have been working in the planning domain by combining task and geometric levels, such as [24, 3]. Further, to guide the planning systems in both levels in the manipulation domain, the use of knowledge with assistance of perception has also been proposed, like the Perception and Manipulation Knowledge (PMK) framework [8] that proposes the integration of a vision-based perception system with knowledge to assist the combination of Task and Motion Planning (TAMP) for complex manipulation tasks. Similarly, [26] proposes the use of knowledge for modeling actions.

The use of experiential knowledge was later proposed in [5], integrated with the knowledge-based reasoning framework [26], to enable robots to perform human-level tasks flexibly in varying conditions using a mechanism based on machine learning for adaptation that allows the robots to exchange knowledge between themselves in different environments. However, the question that arises now is how to automatically build the experiential knowledge. To answer the question, planning and adaptation modules are required.

There exists a large corpus of work on skills representation and execution [1, 6, 17]. Two distinct approaches are commonly preferred in order to represent and execute skills; one at the symbolic level [13, 23], the other at the motion level [12, 5]. The former defines skills at a higher level and allows for generalization and planning, while the latter gives more flexibility for an execution-relevant definition of skills.

High-level symbolic representations many times use graph structures and relational representations [21], while alternative methods, such as [15], capture underlying task structures in the form of probabilistic activity grammars. All these approaches give compact descriptions of complex tasks, but they do not consider execution-relevant motion parameters (trajectories, poses, forces) in great detail. For motion-level representation there are several well-established techniques, such as Dynamic Movement Primitives (DMPs) [12] or hidden Markov models [14]. With motion-level encoding, one can investigate or learn different trajectories with a cognitive sense, i.e., the same skill can be represented by various trajectories.

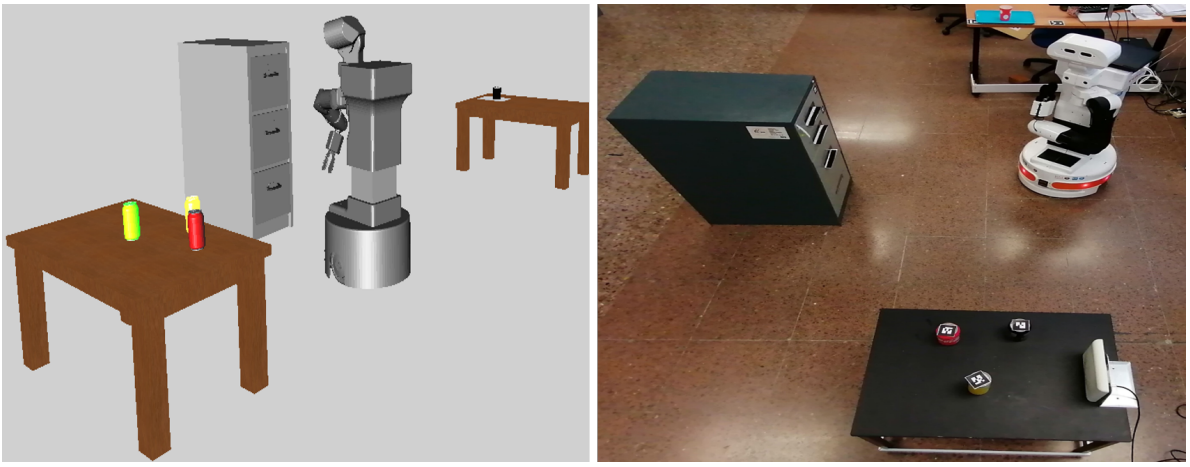


Figure 2: The motivation example. On the left, the represented scene in Kautham Project. On the right, the real scene.

3. Framework

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

3.1. Overview

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. 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. The former includes a task planner to compute a sequence of skills 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 skills 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 skill. This structure is formally defined as a workflow that can be automatically executed through interfacing existing software components of the robot control system. Workflow modeling is devoted to represent the structure of a task and to organize its execution, i.e., the abstract steps required for task execution are described. For example, to execute the *pickUp* skill, a two-step sequence of operations is required: 1) call the Inverse Kinematics (IK) module to check reachability for grasping the object; 2) find a collision-free path towards a grasping configuration. For *putDown* skill, to seek for available placement room to place the object is also required.

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

1. Awareness module, that contains a) Perceptual knowledge to assist the robot to figure out which are the proper algorithms and parameters to be used for available sensors in order to extract data; b) Geometric knowledge to provide the geometric reasoning responsible for checking the feasibility of the skills; 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).

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.
2. geometric issues, like determining if a configuration is collision-free or if an inverse kinematic solution exists

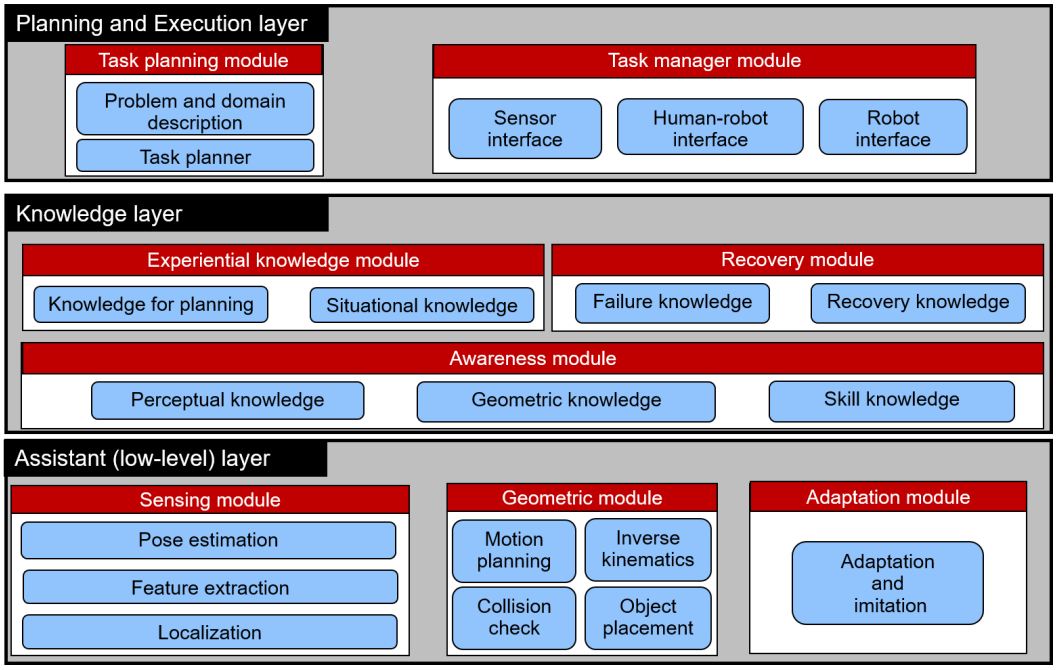


Figure 3: The proposed framework: layers and modules.

for a gripper pose, which are dealt by the geometric module.

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. 3.2 the assistant (low-level) modules used in the assistant layer will be described. Then, in Sec. 3.3 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. 3.4 presents the task planning and task manager modules proposed in the planning and execution layer. Finally, in Sec. 3.5 the framework flowchart illustrates how the proposed layers and their modules are integrated.

3.2. Assistant modules

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

3.2.1. Sensing module

Our 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, 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).

RFID technology

An RFID technology is composed of three main parts: reader, tags and antenna. The tags, like the ones used in [7], 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 [7, 16]. 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 (see Sec. 3.3.5).

The purpose of using RFID in SkillMaN is:

1. 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.
2. 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.

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 [8].

The tags are used to identify the world entities and semantically link them to 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.)

3.2.2. Geometric module

The geometric module provides several services that help a planner to evaluate the feasibility of the skills. It is composed of 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 motion,
3. Motion Planning (MP) used to generate a sampled-based motion 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.

3.2.3. 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 humans do, i.e., we intuitively know how to perform the motion primitives, although our exact motions 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.

3.3. 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 the knowledge can efficiently mange 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.

3.3.1. Experiential knowledge module

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

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

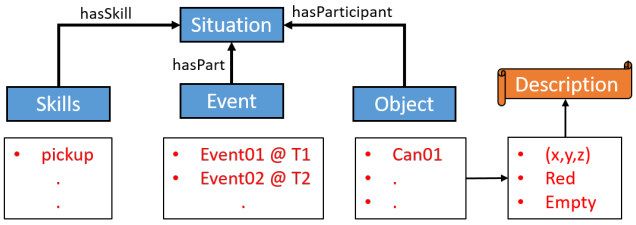


Figure 4: The situation modeling in SkillMaN. The blue and orange are abstract concepts, while the red referred to the instances of the classes. The multi-sensory module is used to build the description of the environment.

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.

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. 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 [18] 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 [8].

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. 3.3.5

3.3.2. 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.

Perceptual knowledge

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. 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 [8], 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, 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

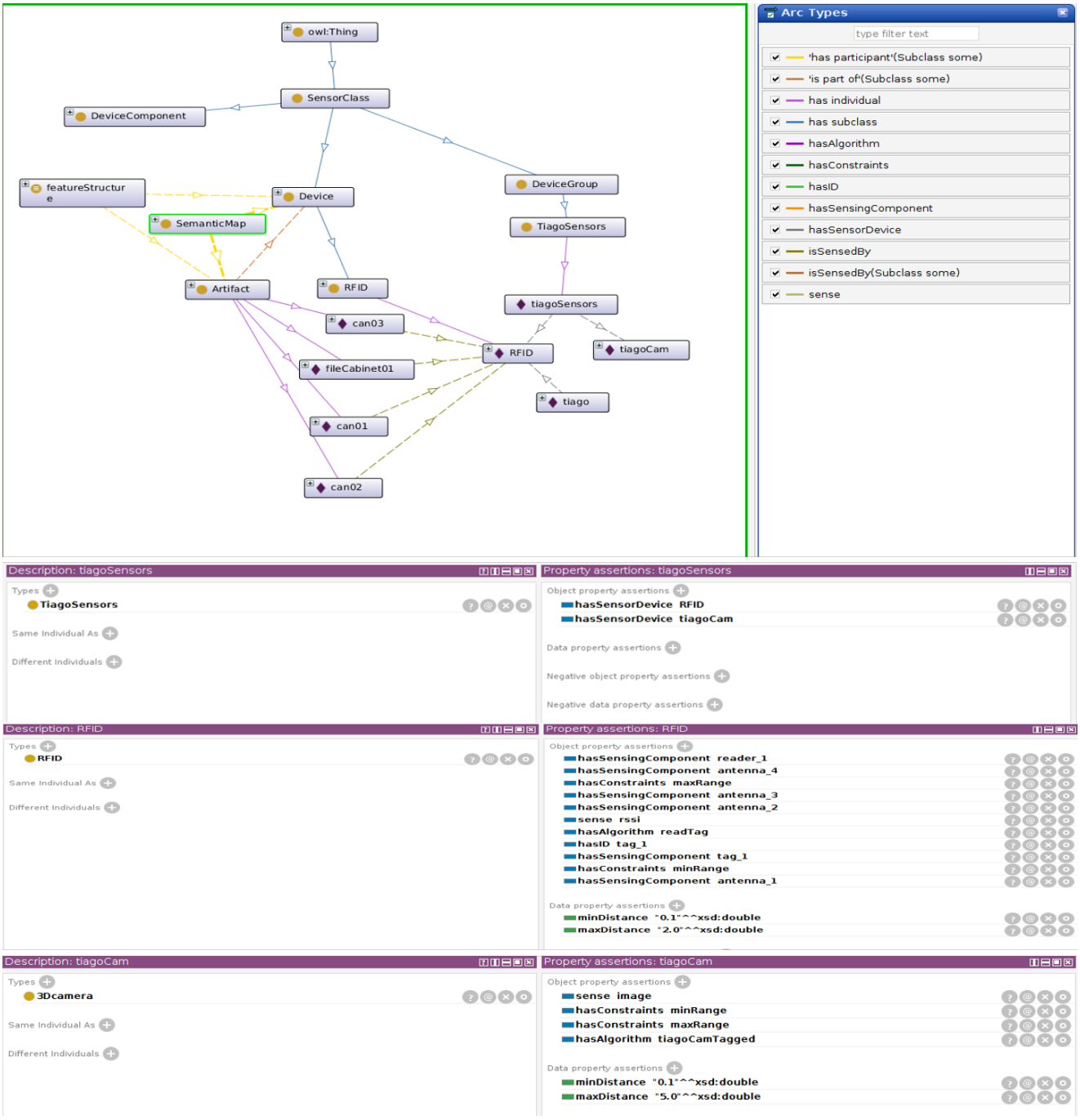


Figure 5: The representation of perceptual knowledge in SkillMaN.

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, [4]), the knowledge of RFID sensors is expressed as:

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

And the knowledge of camera is expressed as:

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

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.

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 [20]:

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.

3.3.3. 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 [9, 10] 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. This part is out of the paper scope and it has been introduced in detail in [9].

3.3.4. 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.

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 [27] 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].
```

Situation similarity check:

The similarity of scenes is computed using taxonomic information from an 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. 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 [28], although for numerical stability reasons, the logarithm of this similarity score is used here, i.e., for two individuals x and y:

$$\text{Sim(x, y)} = \log \frac{\text{depth (lca C(x), C(y))}}{0.5(\text{depth C(x)}) + \text{depth C(y))}} \quad (1)$$

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

a) b)

Figure 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 (storage task).

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 a few 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.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.

Skill reasoning

Symbolic skill reasoning to call the primitive skills from the skill knowledge is used to load the problem and domain description to the task planner. 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.
```

Another question arises is, *"what is the problem?"*, which using a Prolog predicate is:

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

Symbolic skill reasoning using rule-based logic is used to check the satisfaction of the constraints of a situation. For example, to place a can inside the second drawer of a file cabinet requires that the first drawer be closed, i.e., it is one of the preconditions that must hold to be able to execute the place skill in the second drawer. The purpose of inference in this level is to answer 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 (storage task), where the RFID tag memory is used to know the status of the drawer (i.e., open or closed).

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 [8]. 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?*)..

Geometric reasoning:

The main role of geometric reasoning is to evaluate geometric conditions of symbolic skills. 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 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].
```

3.3.5. Data management

Different ways to store 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 a database (DB) and knowledge in ontology form.

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.

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. It allows the instance to take the same taxonomy of the *cup* super class.

3.4. 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.

3.4.1. 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.

The first step in defining the initial state is to get the poses (positions and orientations) of all the available objects in the environment. 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.

```
(: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 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).

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.

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. 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 mechanism for the semantic resolution is beyond the scope of this paper. 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, as presented in [9].

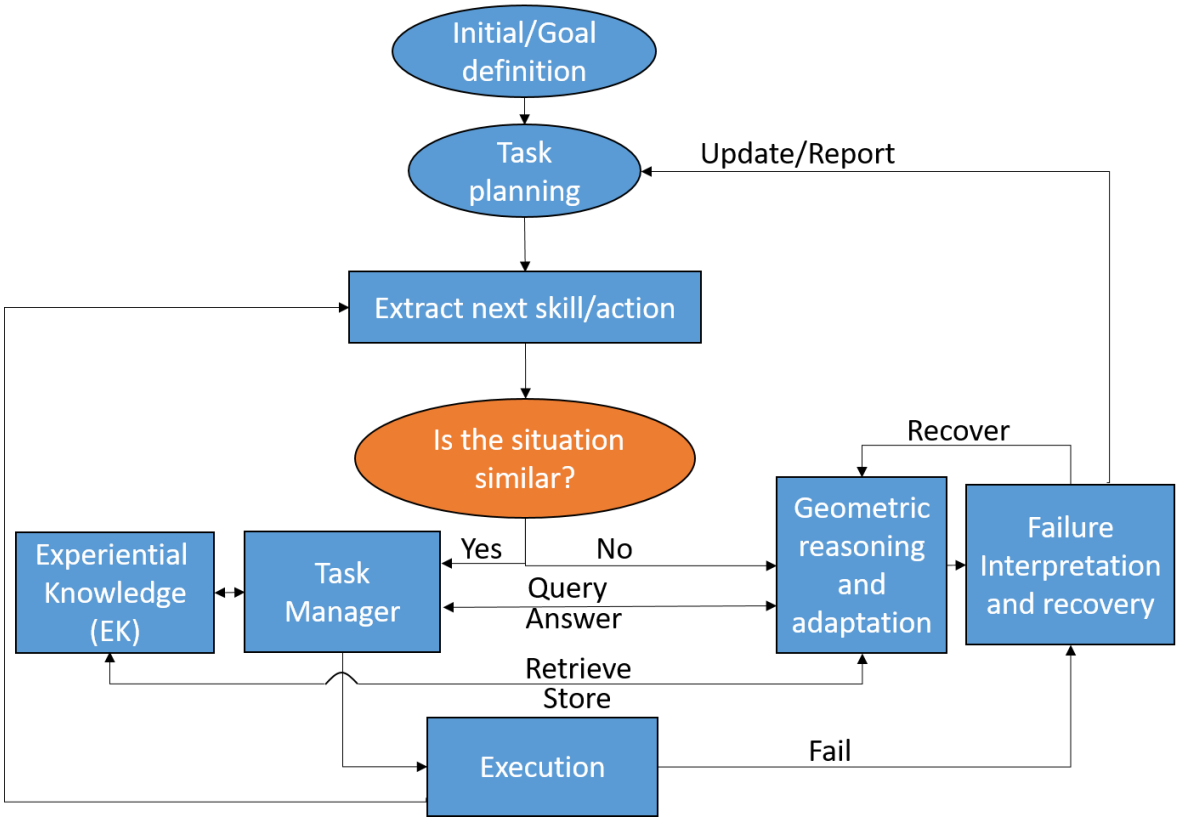


Figure 8: Flowchart of the SkillMaN framework.

3.4.2. 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. 4.2

3.5. Framework flowchart

The flowchart shown in Fig.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 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.

4. Implementation and set-up

4.1. Implementation tools

4.1.1. 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.

4.1.2. 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 [22]. 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) [25] 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 [12]. 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.

4.1.3. 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 framework [8] 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. The reasoning scope of PMK is divided into four parts: reasoning for perception, the reasoning for object features, the reasoning for a situation, and reasoning for planning.

4.1.4. Navigation and mapping

Fig. 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.

Navigation TIAGo has autonomous navigation functionalities implemented using the ROS 2D navigation stack (http:

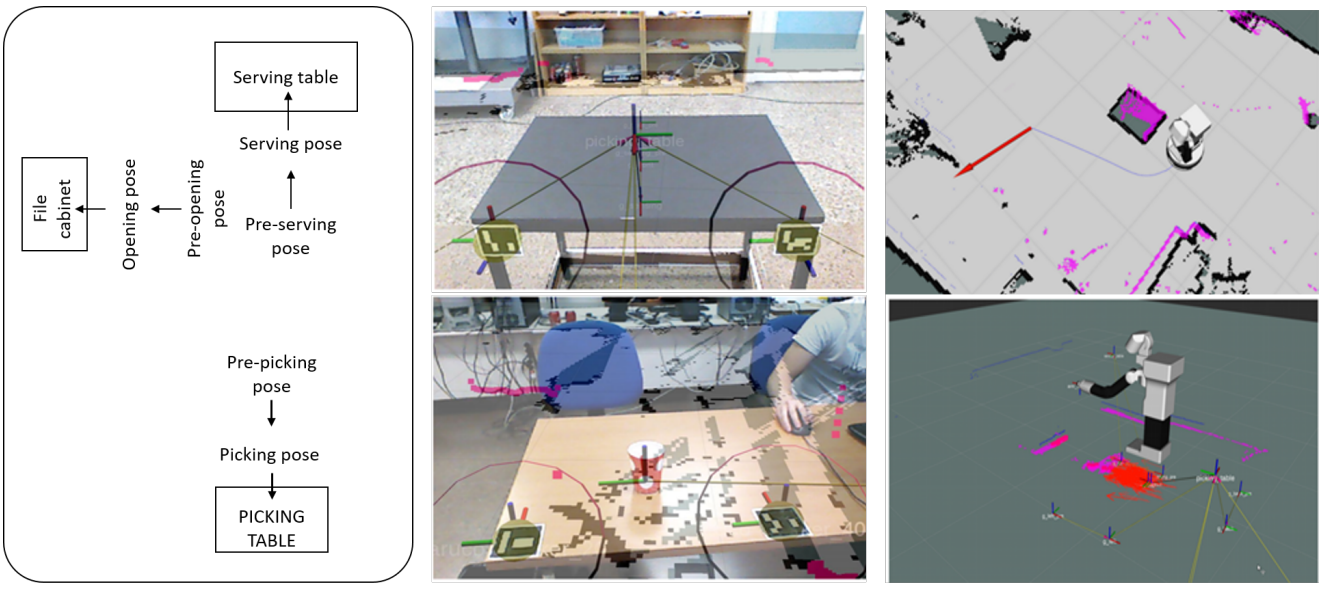


Figure 9: Navigation experiment: (a) the plan view of the indoor environment, (b) the real scene of how the robot detects the tables used in the indoor environment, and (c) path planning, obstacle avoidance capabilities, and navigation poses on the map.

`//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 [11]. 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.

Mapping The mapping and pose generation process starts by creating the occupancy 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.

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.

4.2. 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. 6.3. All the modules mentioned in Fig. 3, and the provided services of each module as described in Fig. 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).

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. 3.3.4
2. *FF* used to automatically compute symbolically the sequence of skills to be executed.

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, i.e., the type of grasp according to the current situation.

The assistant modules of sensing, geometric and adaptation are provided in services form. The sensing services are used to provide the initial state of the environment to the planner or whenever required, using cameras and RFID sensor. 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

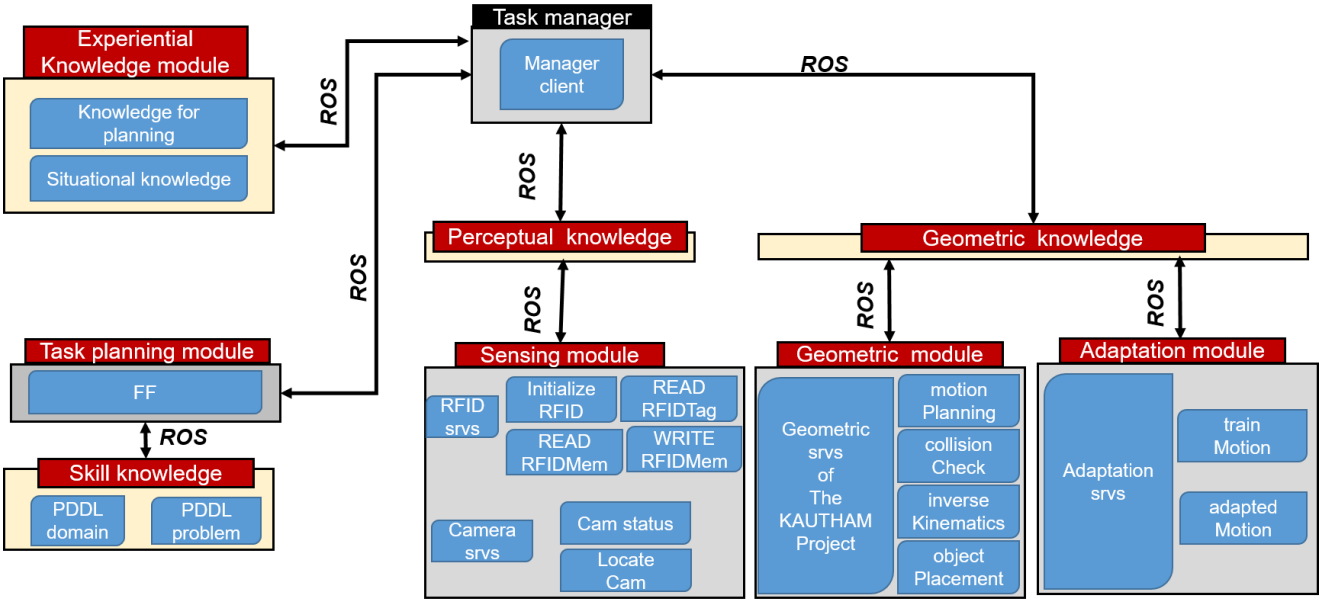


Figure 10: Task management and the communication with the ROS-based services from symbolic and low-level modules.

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             if AdaptedMot = feasible then
16                 execute(AdaptedMot)
17                 store[AdaptedMot, skillName, Y] // store the executed motion, skill and the current scene situation
18         if S = emptySet or C = infeasible then
19             Mot = generateMotion(skillName) // generate a collision-free motion for a new situation
20             if Mot = feasible then
21                 execute(Mot)
22                 store[GenMot, skillName, Y]
23         skillName ← nextAction(P)
```

2. *LocateCam* used to estimate the objects poses and their IDs.

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 sensing module is managed through the perceptual knowledge, the following service is used for this purpose:

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

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.

These 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. 3.3.4), and the collision-free path between them using the *motionPlanning* service.

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. 3.3.4), 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, allowing the task manager to monitor the task that other modules are responsible for.

4.3. Experimental set-up

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

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.

5. 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,
6. All the atomic actions/skills are assumed robust enough to be executed without failures in those scenarios, hence not requiring the use of the recovery module.

5.1. Scenario one: Storage task

Fig. 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. 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. 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. 11-f. The second *can* is full and the corresponding drawer where to be stored is the second one, shown in Fig. 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.

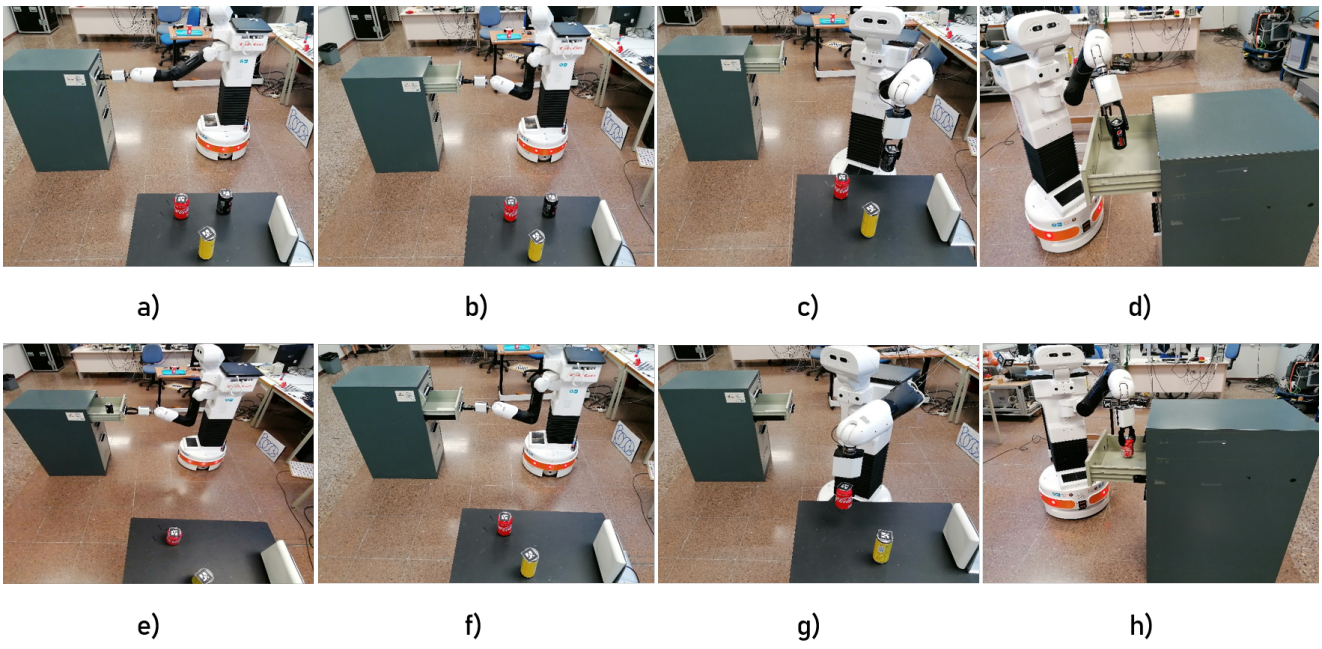


Figure 11: a) the robot plans how to open the first drawer; b) the robot executed the openDrawer skill; c) the robot plans how to pick the black can (based on its status, here it is empty) with the help of experiential knowledge about what is the best grasp to place it in the first drawer; d) the robot executed the place skill; e) the robot executed 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 executed the place skill. Video URL: https://www.youtube.com/watch?v=bTmWAkjC93c

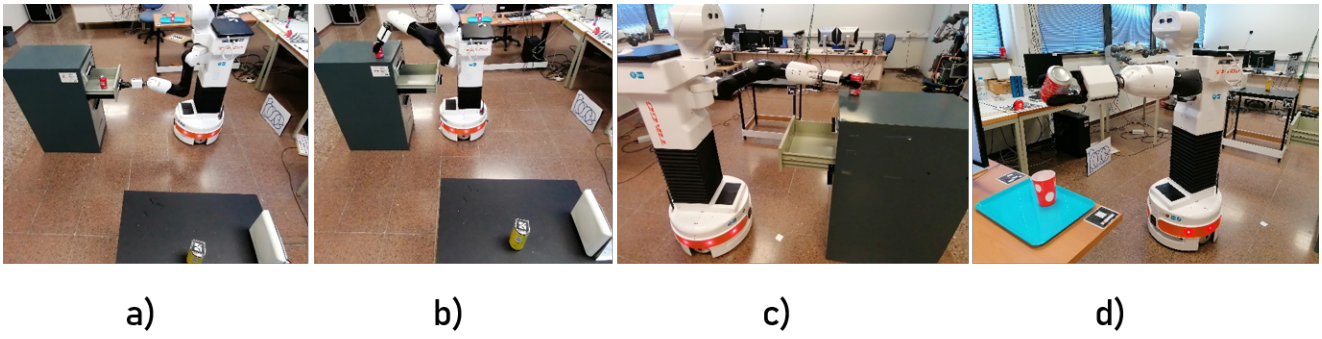


Figure 12: a) the robot checks the similarity of the current situation, it finds the same skill has been used with the same object (i.e., a drawer in the file cabinet), then adapts the skill with the same motion used in the database; b) the robot figures out the top-grasp is not feasible for pouring 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 is serving 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

5.2. Scenario two: Serving task

Fig. 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.

5.3. Result

The perception, planning, and adaptation modules are used with the knowledge for guidance, including experien-

Table 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	–	–

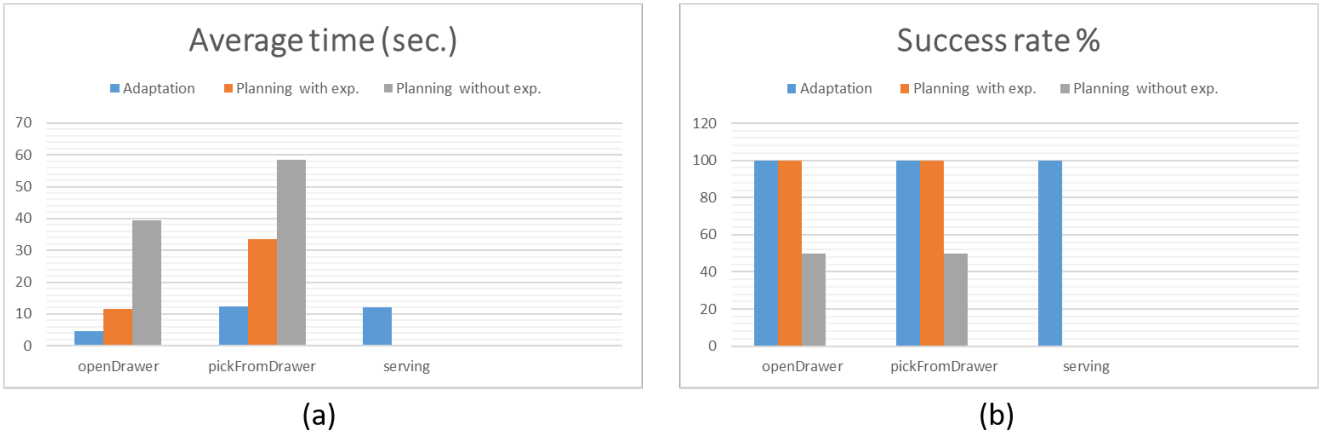


Figure 13: (a) Average time for *openDrawer*, *pickUp from the drawer* and *serving* skills using adaptation and planning with and without experiential knowledge. (b) Success rate for the each skill.

tial knowledge, to successfully execute the proposed tasks. As illustrated in Table. 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 motion 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 motion 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 motion) this is even more relevant.

Due to the aforementioned reason, and as shown in Fig. 13-a, we noticed that the adaptation approach saves time in comparison to repetitively planning similar tasks. We also consider the use of experiential knowledge in the planning in the comparison. Although the planning using experiential knowledge for both *openDrawer* and *pickUp* skills save around 38% of the time without using it, the adaptation method still has better results. Meanwhile, the use of experiential knowledge saves around 29% of the time without using the experiential knowledge for the *openDrawer* skill, and more than half (57%) for *pickUp* skill from the drawer. We executed each skill 10 times.

There are 34 situations stored in the PMK ontology. These situations are described through the axioms and properties of PMK. To describe both aforementioned scenarios, 1254 axioms are used to relate instances, data, and classes. 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. 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.

6. Discussion

6.1. Discussion about the result

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. 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 motion 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 motion.

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 Table. 1, 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 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. 14.

As demonstrated in Fig. 13, and based on experimental evaluations illustrated in Table. 1, the use of the adaptation method is powerful for every-day tasks when a motion already exists in the DB. It is demonstrated in the calling of the collision check module for feasibility verification and execution time.

6.2. Challenges

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 environment as presented in the proposed scenarios or a fully unstructured environments as introduced in our previous works [2, 8].

6.3. System flexibility

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

1. use untagged-based perception systems, such as the one presented in [13],
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 [19], and

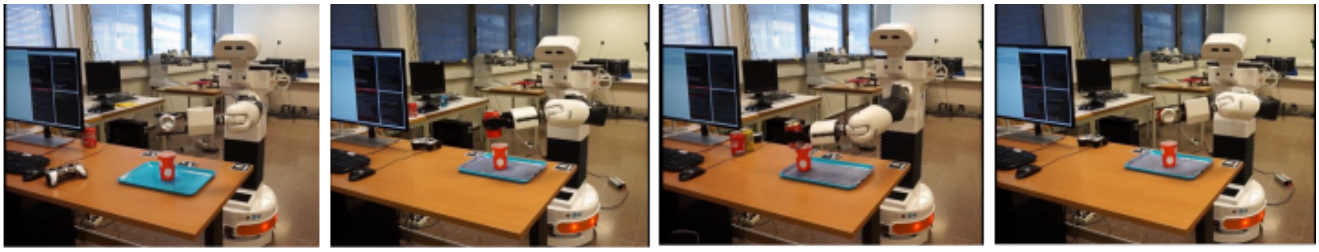


Figure 14: After applying the similarity check module and conclude the current situation exists in the knowledge database, the adaptation process with different poses for serving skill is used.

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).

7. Conclusion and future work

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. A set of skills such as *pickUp, putDown, openDrawer* and *serving* are introduced. The 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. The system shows flexibility to be adapted in several environments and robots. Also, more skills can be added to the framework.

Future work will be focused on increasing 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 to be used in the current situation. Moreover, efforts will be put on increasing the adaptation capability by generating motion templates from humans or planners that could be efficiently adapted in several situations.

References

[1] Aein, M.J., Aksoy, E.E., Wörgötter, F., 2019. Library of actions: Implementing a generic robot execution framework by using manipulation action semantics. The International Journal of Robotics Research 38, 910–934. doi:10.1177/0278364919850295.

[2] Akbari, A., Diab, M., Rosell, J., 2020. Contingent task and motion planning under uncertainty for human–robot interactions. Applied Sciences 10, 1665.

[3] Akbari, A., Lagriffoul, F., Rosell, J., 2018. Combined heuristic task and motion planning for bi-manual robots. Autonomous robots , 1–16doi:10.1007/s10514-018-9817-3.

[4] Baader, F., Horrocks, I., Lutz, C., Sattler, U., 2017. An Introduction to Description Logic. Cambridge University Press, United Kingdom.

[5] Bozcuoğlu, A.K., Kazhoyan, G., Furuta, Y., Stelter, S., Beetz, M., Okada, K., Inaba, M., 2018. The exchange of knowledge using cloud robotics. IEEE Robotics and Automation Letters 3, 1072–1079. doi:10.1109/LRA.2018.2794626.

[6] Calinon, S., Guenter, F., Billard, A., 2007. On learning, representing, and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37, 286–298.

[7] Deyle, T., Reynolds, M.S., 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, pp. 2579–2586. doi:10.1109/IROS.2014.6942914.

[8] Diab, M., Akbari, A., Ud Din, M., Rosell, J., 2019. PMK - A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation. Sensors 19. doi:10.3390/s19051166.

[9] Diab, M., Pomarlan, M., Beßler, D., Akbari, A., Rosell, J., Bateman, J., Beetz, M., 2020a. An ontology for failure interpretation in automated planning and execution, in: Silva, M.F., Luís Lima, J., Reis, L.P., Sanfeliu, A., Tardioli, D. (Eds.), Robot 2019: Fourth Iberian Robotics Conference, Springer International Publishing, Cham. pp. 381–390.

[10] Diab, M., Pomarlan, M., Borgo, S., Beßler, D., Rosell, J., Beetz, Bateman, J., Beetz, M., 2020b. FailRecOnt - An Ontology for Failure Interpretation and Recovery in Automated Planning and Execution, in: Submitted to ECAW2020, Bolzano, Italy.

[11] Durrant-Whyte, H., Bailey, T., 2006. Simultaneous localization and mapping: part i. IEEE Robotics Automation Magazine 13, 99–110. doi:10.1109/MRA.2006.1638022.

[12] Ijspeert, A.J., Nakanishi, J., 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), IEEE. pp. 1398–1403.

[13] Konidaris, G., Kaelbling, L.P., Lozano-Perez, T., 2018. From skills

to symbols: Learning symbolic representations for abstract high-level planning. Journal of Artificial Intelligence Research 61, 215–289.

[14] Lee, D., Nakamura, Y., 2006. Stochastic model of imitating a new observed motion based on the acquired motion primitives. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems , 4994–5000.

[15] Lee, K., Su, Y., Kim, T.K., Demiris, Y., 2013. A syntactic approach to robot imitation learning using probabilistic activity grammars. Robotics and Autonomous Systems 61, 1323–1334.

[16] Li, Y., Mac Namee, B., 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) .

[17] Liu, Y., Li, Z., Liu, H., Kan, Z., 2020. Skill transfer learning for autonomous robots and human-robot cooperation: A survey. Robotics and Autonomous Systems 128, 103515. doi:doi.org/10.1016/j.robot.2020.103515.

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

[19] Moll, M., Muhayyuddin, Kavraki, L., Rosell, J., 2018. Randomized physics-based motion planning for grasping in cluttered and uncertain environments. IEEE Robotics and Automation Letters 3, 712–719.

[20] Munawar, A., De Magistris, G., Pham, T.H., Kimura, D., Tatsubori, M., Moriyama, T., Tachibana, R., 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), IEEE. pp. 527–534.

[21] Pardowitz, M., Knoop, S., Dillmann, R., Zollner, R.D., 2007. Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37, 322–332. doi:10.1109/TSMCB.2006.886951.

[22] Rosell, J., Pérez, A., Aliakbar, A., Muhayyuddin, Palomo, L., García, N., 2014. The kautham project: A teaching and research tool for robot motion planning, in: Proceedings of the IEEE Emerging Technology and Factory Automation (ETFA), pp. 1–8.

[23] Simmons, R.G., Apfelbaum, D., 1998. A task description language for robot control. Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190) 3, 1931–1937 vol.3.

[24] Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., Abbeel, P., 2014. Combined task and motion planning through an extensible planner-independent interface layer, in: 2014 IEEE international conference on robotics and automation (ICRA), IEEE. pp. 639–646.

[25] Şucan, I.A., Moll, M., Kavraki, L.E., 2012. The Open Motion Planning Library. IEEE Robotics & Automation Magazine 19, 72–82.

[26] Tenorth, M., Beetz, M., 2017. Representations for robot knowledge in the knowrob framework. Artificial Intelligence 247, 151 – 169. doi:doi.org/10.1016/j.artint.2015.05.010. special Issue on AI and Robotics.

[27] WIELEMAKER, J., SCHRIJVERS, T., TRISKA, M., LAGER, T., 2012. SWI-Prolog. Theory and Practice of Logic Programming 12, 67–96. doi:10.1017/S1471068411000494.

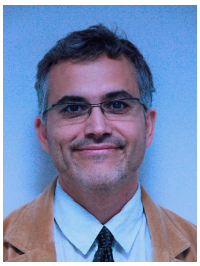
[28] Wu, Z., Palmer, M., 1994. Verbs semantics and lexical selection, in: Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 133–138. doi:10.3115/981732.981751.

Mohammed Diab is currently working toward the Ph.D. degree in Automatic Control, Robotics and Computer Vision at the Institute of Industrial and Control Engineering (IOC), UPC. He is a voting member of the "Autonomous Robots" sub-group of the IEEE WG ORA that works on the standardization of ontologies for robotics and automation. His current research interests are focused upon task and motion planning, perception and knowledge representation and reasoning in manipulation human-robot and robot-robot collaboration, and learning in robotic system.

Mihai Pomarlan is a postdoc researcher at the University of Bremen. His current research interests are focused on knowledge engineering and embodied cognition, with a focus on applications for autonomous service robotics.

Daniel Beßler is a Ph.D. student at the Institute for Artificial Intelligence at the University of Bremen, Germany, where he has also received his Diplom degree (equiv. M.Eng.) in 2014. His main research interest is knowledge representation and reasoning in the context of autonomous robots, and as part of integrated robot control systems. Since 2015, he is the lead developer of the KnowRob knowledge processing system which is the most widely used KR system for service robots. Since 2018, he is a voting member of the "Autonomous Robots" sub-group of the IEEE WG ORA that works on the standardization of ontologies for robotics and automation. Since 2018, he is also a member of the Medical Informatics Research Unit (MIRU) which is a corporation between UNIHB and Mahidol University, Bangkok, Thailand.

Aliakbar Akbari is a Postdoc researcher in Department of Computer Science at Royal Holloway University of London. He did his Ph.D. in Robotics at Universitat Politecnica de Catalunya (UPC). His research interests include robot manipulation planning, Artificial Intelligence, and learning techniques for human-robot Interactions.



Jan Rosell received the BS degree in Telecomumunication Engineering and the Ph.D. degree in Advanced Automation and Robotics from the Universitat Politécnica de Catalunya (UPC), Barcelona, Spain, in 1989 and 1998, respectively. He joined the Institute of Industrial and Control Engineering (IOC) in 1992 where he has developed research activities in robotics. He has been involved in teaching activities in Automatic Control and Robotics as Assistant Professor since 1996 and as Associate Professor since 2001. His current technical areas include knowledge-based task and motion planning, mobile manipulation and robot co-workers.

John Bateman is Professor of Applied Linguistics in the Linguistics and English Departments of the Faculty of Linguistics and Literary Sciences at the University of Bremen. He received his PhD in Artificial Intelligence from Edinburgh University in 1986. His current research areas revolve around multimodal and multilingual semiotic descriptions, formal and linguistic ontologies, functional and computational linguistics, focusing particularly on accounts of register, genre, functional variation, functional natural language semantics, lexicogrammatical description and theory, and computational instantiations of linguistic theory. He has published widely in all these areas, as well as authoring several introductory and survey articles on social semiotics, natural language generation, and film and static document analysis. He is currently the coordinator of the Bremen Transmedial Textuality Research Group (http://www.fb10.uni-bremen.de/bitt).

Michael Beetz received his diploma degree in Computer Science with distinction from the University of Kaiserslautern. His MSc, MPhil, and PhD degrees were awarded by Yale University in 1993, 1994, and 1996 and his Venia Legendi from the University of Bonn in 2000. Michael Beetz was a member of the steering committee of the European network of excellence in AI planning (PLANET) and coordinating the research area "robot planning". He is associate editor of the AI Journal and the coordinator of the German collaborative research centre EASE (Everyday Activity Science and Engineering, since 2017). His research interests include plan-based control of robotic agents, knowledge processing and representation for robots, integrated robot learning, and cognitive perception. In 2019, he received a honorary degree from the University of Örebro for his longstanding cooperation and exceptional, international research.