TU WIEN Informatics

Wissensbasierte Robotikplattform zur autonomen Fertigung

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Ing. Timon Höbert, BSc.
Matrikelnummer 1427936

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Mitwirkung: Dipl.-Ing. Dr.techn. Munir Merdan
Dipl.-Ing. Dr.techn. Wilfried Lepuschitz

Wien, 19. März 2021

Timon Höbert	Markus Vincze

TU WIEN Informatics

Knowledge-Driven Robotics Platform for Autonomous Assembly

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Ing. Timon Höbert, BSc.
Registration Number 1427936

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Assistance: Dipl.-Ing. Dr.techn. Munir Merdan
Dipl.-Ing. Dr.techn. Wilfried Lepuschitz

Vienna, 19th March, 2021

Timon Höbert	Markus Vincze

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Ing. Timon Höbert, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 19. März 2021

Timon Höbert

Acknowledgements

I would like to thank my supervisor, Prof. Markus Vincze, for his support and enabling this thesis. Also, I would like to thank Dr. Munir Merdan and Dr. Wilfried Lepuschitz from the Practical Robotics Institute Austria. All of their feedback and the various discussions helped me to improve not only this thesis but supported the growth of my professional, academic and personal skills. Valuable contributions to this project were also provided by my colleagues Erhard List, Manuel Kisser and Sarah Breit. Finally, every research project benefits from real-world input which I was lucky to receive from our industrial partners, Eric Dokulil, Babitsch Mechanics and Tele Haase.

I would like to express my sincere thanks to Michael Martinides, who first introduced me to programming and nurtured my enthusiasm. Likewise, Dr. Gottfried Koppensteiner encouraged my turn towards robotics and played a decisive part in the development of my passion for science.

I am thankful to have the possibility to conduct research in a field about which I am passionate. I highly appreciate the support of the Vienna University of Technology, my professors and my mentor within the Bachelor with Honors program, Prof. Walter Kropatsch. Being aware of the supporting social network that enabled my studies I am hopeful that the general public will benefit from my future work.

I appreciate the financial support of the "Production of the Future" program by the Austrian Ministry for Transport, Innovation and Technology (contract no. FFG 858707). My work was also funded by the Internet Foundation Austria program Netidee (contract no. 4647).

Finally, I am grateful to always be able to trust in the support of my family, friends and especially my girlfriend. Even if it sometimes felt as if it was quite a ride, I am glad that I choose this road and am determined to continue on this path.

Kurzfassung

Robotersystemen werden vermehrt für komplexe Fertigungsprozesse und die Fertigung kleiner Losgrößen eingesetzt. Die Programmierung und Konfiguration von Robotern ist jedoch zeit- und ressourcenaufwändig und mit hohen Kosten verbunden, welche insbesondere für kleine und mittlere Unternehmen eine Herausforderung darstellen.

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung eines wissensbasierten Frameworks, das die Limitierungen der aktuellen Technik überwinden und die Agilität und Autonomie von Robotiksystemen erweitern soll. Die Fähigkeiten des Frameworks umfassen sowohl logische Schlussfolgerungen als auch Objekterkennungs- und Lokalisationsfähigkeiten. Dies wird erweitert um die Fähigkeit, Prozesse zu planen, geeignete Aktionen auszuwählen und diese schlussendlich auszuführen. Das eingesetzte Produktmodell in Form von Ontologien ermöglicht, dass das Framework Sensordaten semantisch mit Produktmodellen, Handhabungsvorgängen und benötigten Werkzeugen verknüpfen kann.

Das entwickelte Framework ermöglicht roboterbasierten Produktionssystemen eine einfachere Anpassung an die individualisierte Produktion mit kleinen Losgrößen und einer signifikanten Anzahl von Produktvarianten, welche eine schnellere Konfiguration und effiziente Planung erfordert. Der vorgestellte Ansatz wird in einer Laborumgebung anhand einer industriellen Pilotanlage evaluiert.

Abstract

There is a trend to apply robotic systems for small batch production as well as for complex manufacturing processes. However, programming and configuration of robots are time and resource consuming being also accompanied by high costs that are especially challenging for small- and medium-sized enterprises.

The thesis focuses on the development of a knowledge-driven framework, which should overcome the limitations of the state-of-the-art robotics solutions and enhance the agility and autonomy of industrial robotics systems using ontologies as a knowledge-source. The framework includes reasoning and perception abilities as well as the ability to make plans, select appropriate actions, and finally execute these actions. The introduced product model in form of ontologies enables that the framework can semantically link perception data to product models and consequently with handling operations and required tools.

The developed framework enables robot-based production systems easier adaption to individualized production with small lot sizes and a significant number of product variants, which is requiring faster configuration and efficient planning. The presented approach is evaluated in a laboratory environment with an industrial pilot test case.

Contents

CHAPTER 1

Introduction

1.1 Motivation & Problem Statement

Assembly operations are considered to be among the most intensive and costly processes in production lines. This is mainly due to the variability of the assembled parts and the complexity of the executed tasks, with the issue of rising complexity when the demand fluctuates in terms of volume and product types [AGPC16]. The assembly of manufactured goods accounts for over 50% of total production time and 20% of total production cost. In the automotive industry, 20-70% of the direct labor costs are spent on assembly. These statistics show the vital importance of assembly and indicate the prospective savings to be achieved by improving assembly approaches and technologies [EE16]. Cyber-Physical Systems (CPS) are regarded as a key technology for the development of future industrial systems by the strong integration and tight coupling of computational and physical capabilities [DL17].

Today's manufacturing and assembly systems have to be flexible to adapt quickly to large-scale fluctuations of demand, increasing product variants with specific configurations, random dispatching of orders, short delivery lead times of products, and short product life cycles. To compete in global markets, the production lines should be able to effortlessly adapt to new production circumstances without sacrificing cost-effectiveness or product quality [BBB+11]. Robots can play an essential role to provide this transformation as robotics technology, which can prove high efficiency, precision, and repeatability is regarded as a viable solution to cope with these challenges.

However, beyond repetitive tasks of predefined simple pick-and-place actions, traditional approaches to planning and control become complex and expensive, when it comes to handling parts with high variability or acting in less structured environments [ILS+19]. Usually, the robot is equipped with a form of Teach Pendant, which the expert robot integrator can use to program positions and motions of the robot arm. In this way,

industrial applications are usually hard-coded for every specific product type and require manual reprogramming in the case of a newly introduced product variant [WVN+19]. Hard-coded in this context means that all relevant parameters (coordinates and angles) that specify executable robot actions are predefined and known in advance. This kind of traditional control concept for industrial robots requires a significant amount of time, sufficient expertise, and programming efforts for every change and adaptation. Besides, this classical approach suffers from the difficulty that the robot behavior should be specified for all thinkable situations and it has only limited possibilities to adapt in case needed for a given situation [JC16]. The demand for more flexibility on the shop floor requires novel approaches that can cope autonomously and fast with a larger amount of product variability.

To achieve the required level of flexibility and to provide customized production lots, industrial robots need to be more autonomous [KAFZ19]. One of the most important instruments for enabling robots to work more autonomously is knowledge. Robots need to possess relevant knowledge about their environment, capabilities, and actions when performing a specific task [AMQP+13, WZL07]. In this context, significant know-how possessed commonly by human experts should be represented in the robotic system. This means to realize the robot's control in a knowledge-enabled manner, where the robot is equipped with the relevant knowledge needed to perform complex tasks [TB13]. Moreover, the knowledge representation is separated from the program logic and can be used for different scenarios, so the robot can retrieve this knowledge for instance for autonomously generating process plans. Besides, programmers can develop programs that are independent of tasks, robots, or environments, which can significantly speed up the realization of new applications [OABK+19]. A vital part of this approach is that the knowledge is represented symbolically in a way that the robot can understand and extract information that is relevant for further actions.

Ontologies are frequently used to capture the conceptual information models and understand the relationships between entities. They are defined as an explicit, formal specification of a shared conceptualization [Gru95]. In this context, they can be used for modeling specific robotics knowledge offering a consistent view of the involved concepts (objects, actions, capabilities, constraints, states, etc.) but also providing semantic relationships between them. The exploitation of semantics and ontologies in the industrial robot domain is regarded to be very prospective [ZAF16, PRK+19, SKKF+19]. However, current developments are rather task-oriented and a comprehensive generic approach is still required for being able to cover diverse product types as well as related manufacturing activities in industrial robotics, incorporating operations, equipment, manipulation, planning, scheduling, etc. Besides, ontologies alone are not enough to enable the autonomy of robotics systems. Particularly the integration of process planning and ontologies is a challenging research topic, where much more attention has to be set on the symbiosis of planning mechanism with formal modeling and reasoning processes. Besides, considering that much of the available information may be irrelevant for the task that the robot aims to achieve, a challenge is to infer the robot's knowledge and select reasonable and useful

information for a given task [AKL16]. Moreover, there are still some difficulties in the mapping between the involved concepts and seamless transformation from the product or part characteristics and geometry into a process plan. In this context, it would be valuable if the approach could reason about manipulation characteristics and constraints, but also integrate sensing resources to handle possible anomalies.

1.2 Contributions

The following contributions of this thesis are provided to improve the current state-of-the-art: The architecture of the framework, the world model in the form of an ontology, the knowledge transformation in the Decision-Making component and the ontology-based stereo vision module.

1.2.1 Framework Architecture

A novel knowledge-based framework architecture is presented aiming to fulfill the mentioned demands. The framework architecture encapsulates the system components and defines the relations between the components. The developed approach simplifies the engineering efforts and provides industrial robots with more flexibility and adaptability when handling increasing product variants.

1.2.2 World Model

A knowledge base with integrated reasoning services is integrated into the framework, which can exploit the information provided by an ontology-based model to link it with individual actions and the required manufacturing equipment. In this thesis, the framework is extended with ontology concepts additionally covering knowledge about manipulation and assembly constraints as well as their relations.

1.2.3 Decision-Making component

Furthermore, the Decision-Making component can automatically query required domain information including entities and their relations, which are relevant for automated sequence plan generation. The presented approach extends my previous research, where the OWL-PDDL Mapping scheme is introduced [HLM19]. This mapping scheme is implemented in the presented framework to enable the automatic knowledge transformation between the knowledge base and the planner. Also, this work is based on the automatic configuration mechanisms of a robotics system [MHLL19, HLLM19].

1.2.4 Ontology-based Stereo Vision System

This is further extended, by also adding a knowledge-based stereo vision module for automatic object recognition that can semantically link perception data with geometric features represented in the knowledge base. This module is able to, on one side to

recognize the targeted object and on the other side to semantically reason about the object's use. For this purpose, a user interface for a simplified stereo-camera setup is also implemented.

1.3 Methodology

The methodology consists of the following four phases:

1. **Technology Review and Requirements Analysis**

   The status analysis will investigate the current state-of-the-art and practices by reviewing recent promising international research. The goal is to obtain a detailed overview in the field of CPSs, knowledge-based systems, ontologies, stereo vision recognition and their application, especially in the industrial robotics domain. The collection of good practice examples should provide important information about reliable methods and technologies.

2. **System Ontology Development**

   The aim of phase 2 is the development of an ontology to describe the production processes as well as the robotics system. The resulting ontology is based on the definition of a taxonomy of manufacturing components, operations and product parts, which contributes to the formalization and understanding of the manufacturing problem. The ontology needs to represent an accurate model of the robot environment including tools, but also other product and sub-part information, as well as, manufacturing process information.

3. **CPS Control Architecture**

   Phase 3 is concerned with the specification of the CPS and the interaction model to support the goal organization that the system needs to accomplish. Based on the outputs of phases 1 and 2, a conceptual architecture of the CPS for controlling the robotic system will be designed. Special attention will be given to shared infrastructure services and interfaces to the low-level control as well as the vision system and its configuration user interface.

4. **Application and Evaluation**

   The fourth phase constitutes testing and evaluating the entire CPS in two steps. In the first step, synthetic data is used to test and validate the developed core component of the system, the Decision-Making component. The agility and performance of the component are studied simulating different scenarios. This simulation offers also the possibility to test different domains much faster than it would be possible to do in a real system since an accelerated execution is possible. In the second test, the whole system is tested and evaluated with a demonstration assembly use-case. To test the capabilities of the framework, different types of products and automation components are involved, such as two types of robots and a conveyor.

1.4 Structure of the Thesis

This thesis is structured as follows: Related works and the state-of-the-art are analyzed in Section 2. The framework architecture, as well as a description of its four contained components (World Model, Decision-Making, Perception, and Execution) are presented in Section 3. The implemented application is presented in Section 4. Moreover, an evaluation of the approach is discussed in Section 5. Finally, Section 6 concludes the thesis with an outlook on future research.

CHAPTER 2

Related Work

To perform particular tasks, an autonomous robotics system needs to perceive the environment, store and retrieve knowledge and plan actions. The necessary knowledge base in the form of Ontologies is outlined in the following Section 2.1. Section 2.2 highlights planning systems and planning languages with a special focus on robotics. Robotics frameworks that incorporate both knowledge bases and planning are listed in Section 2.3. Finally, machine perception methods for object detection are summarized in Section 2.4 .

2.1 Ontologies and the Semantic Web

To perform particular tasks, an autonomous robotics system needs to consider several issues such as actions to perform, involved objects, behaviors and tools to be used as well as how to approach and manipulate objects. Knowledge plays a key role in the creation of relationships between tasks, robot capabilities, perception, and actions. The knowledge is expressed via knowledge representation methods to provide a machine-interpretable model and enable the robot to understand the application domain, where symbols represent objects or concepts from the real world [SZ19].

2.1.1 Semantic spectrum

In general, there are multiple semantic models to formalize knowledge. The spectrum of models can be ordered by their semantic expressiveness, as shown in Figure 2.1. On one side of the spectrum are simple *lists*, like glossaries or dictionaries, which provide only syntactic interoperability by defining a set of terms. On the other side of the spectrum are the strong semantics of ontologies, which can also be used for automated reasoning. Ontologies can be defined as “a formal, explicit specification of a shared conceptualization”. [SBF98].

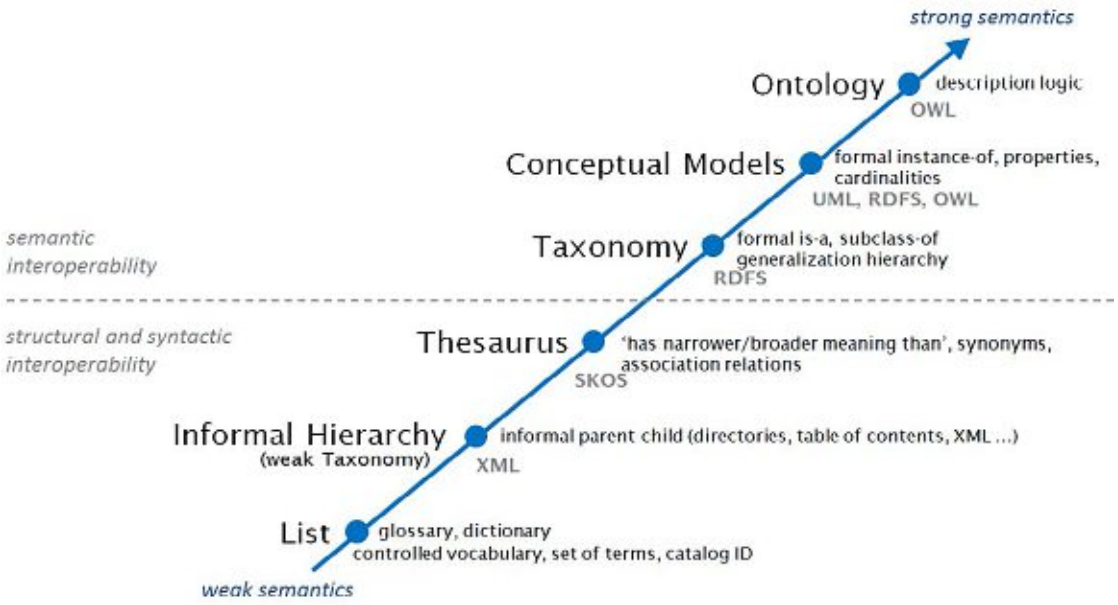


Figure 2.1: The Semantic Spectrum of Knowledge Organization Systems. Image by Geoff Gross.

From left to right on this spectrum the degree of meaning and formality increases which, vice versa, also decreases ambiguity. Additionally, with increasing expressiveness, complexity also increases, which reduces efficiency [SS10]. As shown, there are multiple levels of semantic models in between which build upon each other. There is the *Informal Hierarchy*, which, in addition to a simple list, defines only an informal parent-child relation, as, for example, a directory structure or any representation in the Extensible Markup Language (XML) can formalize. This so-called, *weak Taxonomy* only formalizes the relations semantically arbitrary, for example, in an application-oriented way to serve a specific task [SS10]. On top of that, there is the *Thesauri*, which defines four relations: Equivalence (same-as), Homography (spelled-same), Hierarchy (broader/narrower-than), and Association (related-to). For example, the W3C Standard Simple Knowledge Organisation System (SKOS)[1], can be used to model a Thesaurus [AH11]. On the other hand, the *(Strong) Taxonomy*, formalizes hierarchy in a semantically strict way, to model true subsets with the subclass relation. For example, the standard of RDF Schema (RDFS) defines a formal syntax to formalize a Taxonomy [SS10]. This semantic model is extended with instance-class relations, properties and cardinalities, which are *Conceptual Models*, as, for example, the Unified Modeling Language (UML), RDF Schema (RDFS), and some Web Ontology Language (OWL)-standards represent [McG02, McC04].

[1]SKOS, available at `https://www.w3.org/2004/02/skos/`

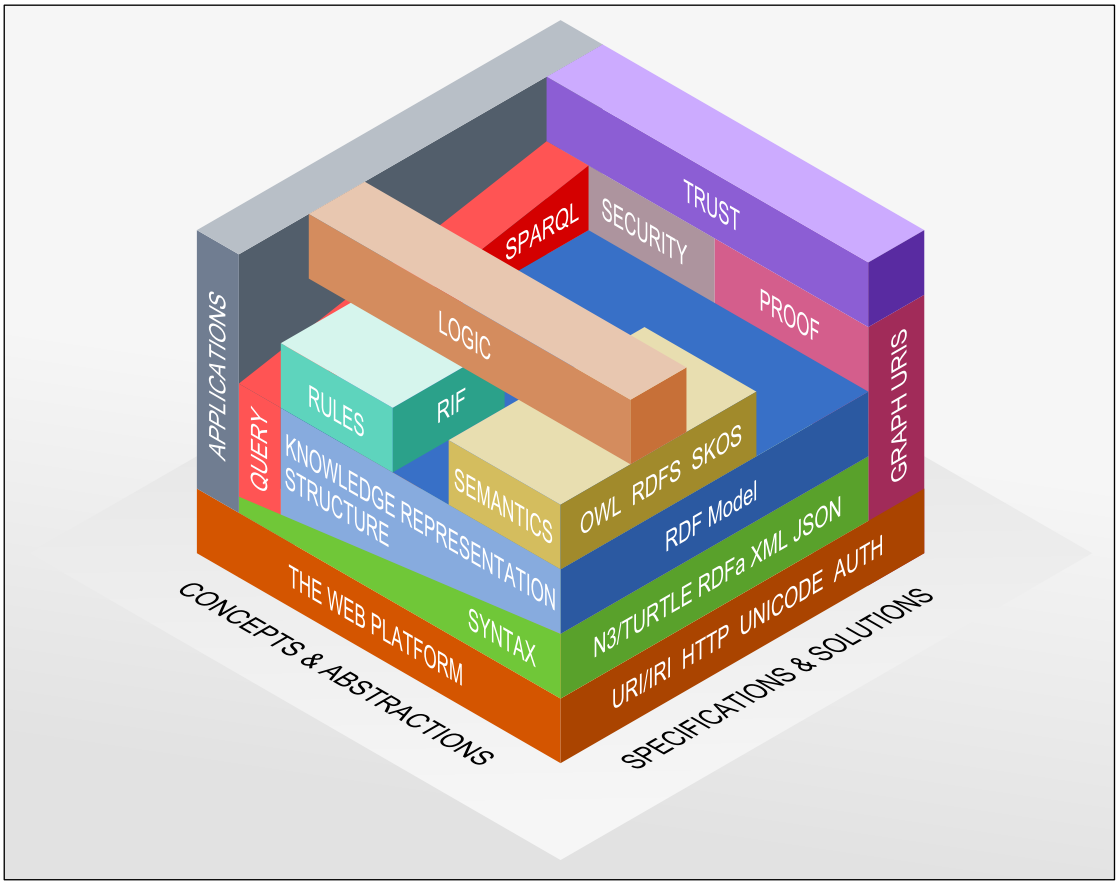


Figure 2.2: The Semantic Web Stack. Image by Benjamin Nowack[2]

2.1.2 Semantic Web Technologies

The Semantic Web describes the concept for the internet of a "universal information space" for not only human users but also machines. One important building block of the semantic web is ontologies as a knowledge base [BLHL01]. The layered architecture stack, as visualized in Figure 2.2, shows the technologies of ontologies in context to their dependent technologies for syntax, knowledge representation, querying, and semantics.

Resource Description Framework Model (RDF)

The Resource Description Framework (RDF) is the foundation of the semantic web [AH11]. It is a modeling language to represent a world. This world can be the real world, or any world of concepts, for example, to model ideas or documents [McC04]. RDF is recommended by the W3C in the current version 1.1[3]. The elementary building blocks

[2]Semantic Web Stack by Benjamin Nowack, available at `http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake`

[3]Resource Description Framework (RDF), available at `https://www.w3.org/2004/02/skos/`

```
1  @prefix pria: <http://pria.at/ont/knowdrift.owl#>
2
3  <http://pria.at/ont/knowdrift.owl#Gripper1> pria:canHandle pria:Part_KL1 .
4  pria:Gripper1 pria:hasSize "42.0^^xsd:double" .
```

Listing 2.1: Two RDF triple examples with a full URI, CURIE and a value.

of RDF are RDF-statements, the RDF triples [SS10]. An RDF triple is defined in the following form:

$$\text{subject predicate object} \tag{2.1}$$

In many cases, this structure is used to relate an entity (subject) by an attribute (predicate) with a value (object) [McC04]. The resulting data structure is a directed, labeled, attributed, semantic multi-graph, where subject and object represent the nodes and predicates the links [SS10]. RDF-resources rely on Uniform Resource Identifier (URI) references. For example, the two triples in Listing 2.1, models the ability of a gripper with size 12 to manipulate a specified part.

Note that the first RDF-resource, the subject, uses the full URI notation, whereas the latter two use the abbreviation scheme Compact URI (CURIE) which is defined in the header [SS10]. In this example, the namespace *pria*, followed by a colon substitutes the long URI *http://pria.at/ont/knowdrift.owl*. There is also a default namespace that is identified without any symbols, by just the colon [AH11]. The use of this abbreviation scheme can enhance readability and serialization file sizes.

There are multiple RDF-syntax representations to serialize RDF-statements. The showed example uses the N3 RDF Syntax[4], which is designed for human readability. N3 itself originates from the Turtle syntax, which is a subset of N3. Traditionally, the original RDF syntax is based on XML, because of the first RDF standardization in 1999. Modern systems also use a representation based on JavaScript Object Notation (JSON) called JSON for Linking Data (JSON-LD)[5].

In many cases, consecutive RDF triples are referencing the same subject. Similarly to the English language, the N3 syntax enables the compact notation of extending an existing triple with a semicolon, directly followed by another predicate and object to reference the same subject. This scheme can also be used with a comma separation to list multiple objects, as shown in Listing 2.2.

Additionally, RDF defines a vocabulary of multiple predefined RDF-resources, for example, *rdf:type, rdf:Property, rdf:List* [SS10].

In RDF applications, it can be necessary to track the origin of triples or to group multiple graphs by an identifier [AH11]. For this purpose, triples can be extended with

[4] N3 RDF Syntax, available at `https://www.w3.org/TeamSubmission/n3/`

[5] JSON-LD Syntax, available at `https://www.w3.org/TR/json-ld11/`

```
1  pria:Gripper1  pria:canHandle   pria:Part_KL1  .
2  pria:Gripper1  pria:canHandle   pria:Part_REL9 .
3
4  pria:Gripper1  pria:canHandle   pria:Part_KL1  ;
5                 pria:canHandle   pria:Part_REL9 .
6
7  pria:Gripper1  pria:canHandle   pria:Part_KL1  ,
8                                  pria:Part_REL9 .
```

Listing 2.2: Three times the same two triples in two N3 abbreviation representations.

an additional attribute, the graph resource, to form a four-tuple. These tuples are also referred to as quads.

Resource Description Framework Schema (RDFS)

RDF Schema (RDFS)[6] builds on top of RDF for additional classification and specification in a similar manner to object-oriented-modeling. For this purpose, the standard specifies additional resources, such as *rdfs:Class, rdfs:subClassOf, rdfs:range, rdfs:domain* to model class and attribute relationships [SS10].

However, RDFS has some limitations [SS10]:

- First, properties, as defined with *rdfs:range*, are fixed for all classes. In other words, a range restriction can not be specified to apply only for some classes.
- Secondly, the disjointness of classes can not be specified with RDFS.
- Vice-versa, the definition of disjoint unions of classes to form new classes is also not possible.
- With RDFS it is not possible to specify any form of cardinality for properties.
- Finally, special characteristics of properties can not be expressed, for example, transitivity, uniqueness, or inverse properties.

To overcome these issues, the Web Ontology Language (OWL) is specified by the W3C.

Web Ontology Language (OWL) and Semantic Reasoning

The Web Ontology Language (OWL)[7] extends the RDF/RDFS-stack with additional expressivity. It was designed to provide a well-defined syntax and semantic, efficient reasoning support, sufficient and convenient expressiveness [SS10]. The formal semantics of additional axioms enables automated reasoning for the following purposes:

[6] Resource Description Framework Schema (RDFS), available at `https://www.w3.org/TR/rdf-schema/`

[7] Web Ontology Language (OWL), available at `https://www.w3.org/TR/owl-features/`

- *Class membership:* Every instance of a class is also an instance of its parent classes.
- *Class equivalence:* Pairwise class equality form a connected equivalence class. For example, if class A equals B and B equals C, we can infer that A also equals C.
- *Consistency:* Errors in the data model lead to impossible states which can be revealed by semantic reasoning. For example, two classes can not be disjoint and equal at the same time.
- *Classification:* OWL enables a definition of sufficient conditions for class membership. This can be used to automatically reason if an instance is a member of this class by checking the defined conditions.

This higher expressivity comes with the costs of higher complexity and more inefficient reasoning. To overcome this issue, OWL is divided into multiple overlapping subsets [SS10]:

OWL Full All language primitives are allowed in any combination. This comes with the benefit of maximum syntactical and semantic compatibility with RDF/RDFS. As a drawback, efficient reasoning is not possible, since it is logically undecidable.

OWL Description Logic (OWL DL) The OWL DL subset restricts the language to Description Logic to enable more efficient reasoning. Among other things, the most important restrictions are the explicit typing declaration for each resource and the partitioning of resources, especially properties. The latter restriction means, that each resource can be either a class, datatype, property, individual or value, but not multiple types at the same time. This comes with the disadvantage of compatibility, since not every RDF document is a valid OWL document. However, the inverse statement is still true: Every OWL document is a valid RDF document.

OWL Lite The restrictions of OWL DL are further extended in OWL Lite by removing the support of arbitrary cardinality, disjointness and enumerated classes. This reduced complexity aims for easier implementations.

Finally, OWL only defines the data format for defining and storing data as a knowledge base. For further processing and querying of this knowledge, the SPARQL Protocol and RDF Query Language (SPARQL) is needed.

SPARQL

The recursively abbreviated SPARQL Protocol and RDF Query Language (SPARQL)[8] defines a query language for RDF data with a syntax related to N3/Turtle. The syntax is

[8] SPARQL Protocol and RDF Query Language (SPARQL), available at `https://www.w3.org/TR/rdf-sparql-query/`

```
1  PREFIX pria: <http://pria.at/ont/knowdrift.owl#>
2
3  SELECT ?gripper
4  WHERE
5  {
6      ?gripper pria:canHandle  pria:Part_KL1 .
7      ?gripper pria:canHandle  pria:Part_REL9 .
8      ?gripper pria:hasSize ?size
9      FILTER (?size > 26)
10 }
```

Listing 2.3: A SPARQL example, to query all grippers which can handle the Part KL1 and REL9.

also related to the Structured Query Language (SQL), with its SELECT-FROM-WHERE pattern [AH11]. The SELECT clause specifies the retrieved data and their order. The optional FROM clause specifies the data source. An unspecified FROM clause represents the default data source of the knowledge system. The pattern in the WHERE clause represents a subgraph template with blank variable nodes which is searched in the graph. It can also define boolean FILTER constraints which need to be satisfied to match.

In the case of multiple graphs in the same knowledge-base, the optional GRAPH subclauses can specify the fourth resource of the quad.

2.1.3 Ontologies in Robotics

As stated before, Ontologies are used as a form of knowledge representation and they are a widely implemented formalism in the industry lately [KGJR+16, SKKF+19]. Ontologies are a powerful solution to capture and to share knowledge by making domain knowledge explicit and enabling machines to reason, but also to meet the requirements needed to create autonomous robotic systems [SPM+12].

The application of ontologies in industrial robotics is in the early stage, but several accomplished examples have used ontological modeling for different purposes. The Institute of Electrical and Electronics Engineers Robotics and Automation Society (IEEE RAS) proposed the Core Ontologies for Robotics and Automation (CORA) standard for knowledge representation and reasoning in the robotics and automation domain. The CORA aims to keep consistency and provide a structure for other sub-ontologies developed for this domain, encompassing concepts such as robots, robotic systems, and robot parts [SPM+12] as well as positions [CFP+13]. Balakirsky et al. extended the CORA ontology and focused on generic and specialized robot task representation for the industrial application [ASM17]. The European project RObot control for Skilled ExecuTion of Tasks (ROSETTA) proposed the ROSETTA core ontology intending to offer a generic domain ontology for industrial robotics which consists of multiple sub-ontologies. The focus was on robotic devices and skills, where every device can provide one or more skills that can realize specific tasks. Skills are compositional items

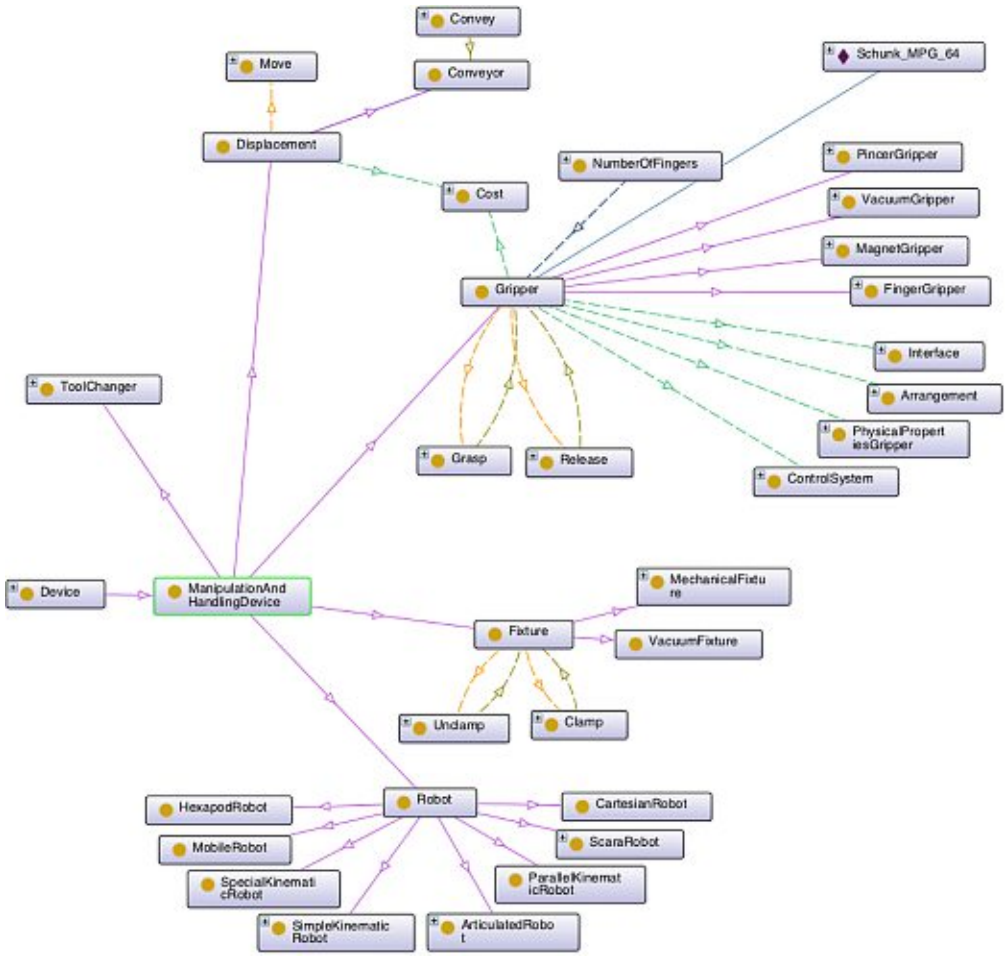


Figure 2.3: Manipulation and Handling device ontology from the ROSETTA project. Image taken from [SM15].

being either primitive (non-divisible) or compound items [SM15]. Figure 2.3 shows an excerpt of the comprehensive data model for manipulation and handling devices. Also, Huckaby and Christensen have presented a taxonomy for assembly tasks in the domain of manufacturing and industrial robotics [HIC13].

Perzylo et al. presented the OntoBREP ontology for Computer-aided design (CAD) data and geometric constraints as a link between product models and semantic robot task descriptions. Computer-aided design (CAD) data is modeled using the Boundary Representation (BREP) of the 3D objects. For this purpose, the authors created an ontology model for topological and geometrical entities, as shown partly in Figure 2.4. The atomic entities of this boundary representation are *vertices*. Two vertices and a curve define an *edge*. Adjacent edges can form a *wire*. A closed wire can define the boundary

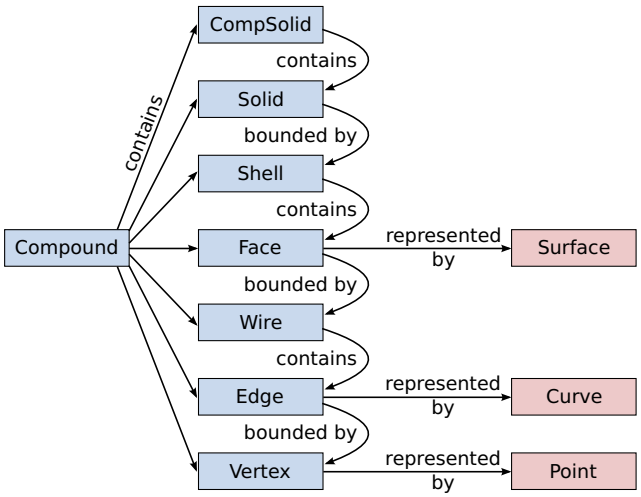


Figure 2.4: Overview of the OntoBREP Ontology classes. Topological entities of the class model are highlighted in gray, geometric entities in red. Image taken from [PSRK15].

of a surface, which is defined as a *face*. Adjacent faces can form a *shell*. If such a shell is defined without any holes, it can be represented as *solid*. To group multiple solids or topological entities, *CompSolid* and *Compounds* can be used. The OntoBREP-based representation can also be utilized to parameterize the task with a set of geometric constraints between the involved assembly parts [PSRK15].

2.2 AI Planning

To handle highly volatile production environments, robots have to raise the level of autonomy and be able to automatically generate effective action sequences, reducing required engineering effort [KAFZ19]. The process of automated reasoning about plans to achieve a given goal is called Planning, which is a branch of Artificial Intelligence [HLMM19] (AI). Nevertheless, currently deployed commercial robotics systems mostly have no or minimal planning capabilities and are typically manually programmed for a specific task [AKL16]. The further application of planning in robotics is also restricted due to the strict coupling of the planning component with the execution component [KP20]. The necessary decoupling can be achieved by Planning languages.

2.2.1 Planning languages

There are many specialized problem-solving algorithms for well-defined domains. The aim of problem description languages is the generalization of domains to a domain-independent representation which can be solved by any planning problem solver. The planning problem solvers, in short *Planners*, take the problem specification as input,

derive the search space and use some heuristics to solve it [HLMM19]. In practice, not every planner can solve every problem, but it enables decoupling of the domain-specific problem implementation and the solver implementation. In other words, the development of the domain-specific problem specification can be done by a domain expert, which only needs minimal knowledge about planning algorithms and the other way around. From a broader perspective, problem description languages are located in between the knowledge representation and the problem-solving technique [HLMM19].

The main purpose of planners is the solving of state transformation problems. Such problems are defined by an initial state, the desired goal and possible actions to change the state. A plan, in this context, is a list of actions that change the state from the initial state to the desired goal state. Haslum et al. lists four different types of planning problems, based on the action model.

1. *Classical Planning* is based on the assumption that the world model is deterministic, discrete and non-temporal. Besides, there is an assumption that the world is static and the planner has complete knowledge of the initial state. In other words, there are no external changes besides the planned actions.
2. *Numeric Planning* also allows the usage of continuous variables for quantification.
3. In *Temporal Planning* the time and timing of actions are also considered by the planner. Additionally, predictably timed events can also influence the state.
4. *Hybrid-system Planning* combines numeric and temporal planning but also includes continuous processes and external events

Even though, some assumptions are unrealistic and limiting, planning based on an approximate model can help to solve more complex real-world problems [HLMM19].

Various research efforts in the area of automated planning have resulted in different approaches to generate a sequence of actions for robots, as shown in Section 2.2.3. In this context, task planning problems are described using different languages such as the Planning Domain Definition Language (PDDL) [FL03], Answer Set Programming (ASP), Hierarchical Task Network (HTN), or the action language BC. Especially PDDL has been a de-facto standard modeling language for automated planning [HLMM19].

2.2.2 Planning Domain Definition Language (PDDL)

Planning Domain Definition Language (PDDL) was introduced in 1998 for the AIPS-98 planning competition (Artificial Intelligence Planning Systems) with the aim to compare the performance of different competing planning algorithms empirically with the same input [GKW+98]. The PDDL syntax is based on the Lisp programming language with its apparent parentheses [HLMM19]. The PDDL is designed to separate the generic domain specification, the PDDL-domain, from the problem-specific data, the PDDL-problem [GKW+98].

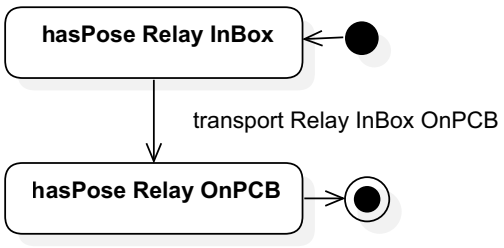


Figure 2.5: The state transitioning diagram of the PDDL pick-and-place scenario.

PDDL-domain

The PDDL-domain defines the state variables in form of predicates and the state transitions, the PDDL actions [HLMM19]. In other words, a PDDL-domain describes the environment where the planner operates. A predicate is a Boolean-typed statement, so it can be either true or false [GKW+98]. From an object-oriented point of view, predicates represent attributes and relationships of and between objects. Predicates can have a variable number of parameters to relate other objects with the predicate [HLMM19]. Predicate parameters are identifiable by the question mark before the parameter name. A PDDL-domain can support datatypes to limit the allowed parameter object types. If datatypes are supported, the PDDL domain can specify a type hierarchy with the root type *object*.

A PDDL action is defined by, at least, its preconditions and its effects. The preconditions of an action define which predicates in the model have to be true to start the action. Predicates can be logically combined with *and*, *or*, etc [GKW+98].

Listing 2.4 shows an example of a PDDL-domain for a pick-and-place environment. As seen, a PDDL-domain definition starts with a heading name of the domain (*PickPlace*). Further on, it is also possible to specify the used language features as *requirements*, for example, *typing*. For each specified PDDL-domain, multiple PDDL-problems can be specified which refer to this domain.

PDDL-problem

A PDDL-problem describes the initial state and the goal the planner should meet, as shown in Listing 2.5 of a PDDL-problem example in the pick-and-place domain. In the beginning, it defines the name of the problem (*PickPlace_1*) and the associated domain (*PickPlace*). Further on, all objects are listed (*:objects*). Since the domain allows typing, for each object the associated data type is specified.

The solution for the stated problem is a plan with only one action: *(move Relay InBox OnPCB)*. The state transitioning graph is visualized in Figure 2.5

```
(define (domain PickPlace)

    (:requirements :strips :typing)

    (:types
        Pose PhysicalObject - object
        Part - PhysicalObject
    )

    (:predicates
        (hasPose ?physicalObject - PhysicalObject ?pose - Pose)
        (canBeMoved ?object - PhysicalObject)
    )

    (:action move
       :parameters (?part - PhysicalObject ?from - Pose ?to - Pose)

       :precondition (and (hasPose ?part ?from)
                          (canBeMoved ?part)
                     )

       :effect (and (hasPose ?part ?to)
                    (not (hasPose ?part ?from))
                )
    )
)
```

Listing 2.4: A PDDL-domain example that defines a robotics pick-and-place environment. There are two types of objects, *Pose* and *PhysicalObject*, whereas *PhysicalObject* has a subtype Part. It defines two predicates *hasPose* and *canBeMoved*, which have one and two typed parameters, respectively. Finally, one action is listed (*move*), which can only be executed if the part to move is movable (*canBeMoved*) and at the *from* pose. If executed, the previous pose *from* is unset for the part and the new pose *to* is set.

```
(define
        (problem PickPlace_1)
        (:domain PickPlace)
        (:objects
            Relay - Part
                InBox OnPCB - Pose
        )
        (:init
            (hasPose Relay InBox)
                (canBeMoved Relay)
        )
        (:goal (and
                (hasPose Relay OnPCB)
        ))
)
```

Listing 2.5: A PDDL-problem example that defines a pick-and-place scenario. A part *Relay* needs to be moved from its initial pose *InBox* to its goal pose *OnPCB*. A part can be attributed with the predicate movable *canBeMoved* and the predicate *hasPose*, which states the current pose of the part.

PDDL versions

Since the first Version of 1998 multiple updating specifications were introduced to integrate new language features. On the other hand, different language aspects were also removed because of the lack of usage or planner implementations [HLMM19].

1. *PDDL 1.0* In 1998, the first version of PDDL formalized classical planning [GKW+98].
2. *PDDL 1.2* Two years later, the second version removed unused language features to simplify the language [Bac00]
3. *PDDL 2.1* This major version introduced numerical and temporal planning language features. For that purpose, it enabled the specification of an objective function for optimization [FL03].
4. *PDDL+* The same authors, extended the language to allow hybrid planning [FL06].
5. *PDDL 2.2* This version added strict axiom definition possibilities, as well as predictable event scheduling definitions for temporal planning [EH04].
6. *PDDL 3.0* Temporal planning is further enhanced by the possibility to specify temporally extended goals and preferences [GHL+09].
7. *PDDL 3.1* This version allows the definition of action costs for objective optimization, as well as finite state variables (object fluents). [HLMM19].

2.2.3 Planning in Robotics

PDDL is widely used as a standardized planning language for a variety of planning actions for robots to reach specific goals [JZKS18]. An approach that is focused on the development of a modeling concept, which enables realistic automated planning and scheduling in discrete manufacturing is presented by Rogalla et al. [RFN18]. Kootbally et al. investigated the idea of automating plan generation using PDDL when applied in a simulation environment. Once formulated, a PDDL plan is combined with knowledge from a MySQL database to form a sequence of executable low-level commands [KSL+15]. Wally et al. presented a model-driven approach for the automated (re-)generation of production plans from a production system model, which describes the available production resources, their capabilities as well as material to be used [WVN+19]. Finally, Pedersen and Volker presented an approach of automated planning in industrial logistics, which utilizes an ad-hoc World Model based on robot skills and their parameters [PK15]. However, it is still a significant shortcoming that the domain knowledge in the existing planners is generated manually, and approaches for the automatic generation of domain knowledge as well as the fusion of existing knowledge systems are required [SZ19].

2.3 Knowledge Frameworks

To support robot autonomy, several Knowledge Frameworks have been developed in research projects, which use reasoning mechanisms and include knowledge sources but are also able to provide planning.

- The Knowledge Integration Framework (**KIF**) is a repository that contains robotic ontologies (e.g. ROSETTA), data repositories, and a reasoning mechanism. The *KIF* provides also services for the Engineering System, which is a robot programming environment, and the robot Task Execution system that generates the run-time code files [SM15].
- **KnowRob** is a knowledge processing infrastructure for cognition-enabled robots. It incorporates components for knowledge acquisition, automated reasoning, visualization, and querying for information. The *KnowRob* knowledge processing system is also integrated with a robot's control program and perception components [TB13].
- The Perception and Manipulation Knowledge (**PMK**) framework aims to enhance task and motion planning capabilities in the manipulation domain. It combines an ontology framework and reasoning mechanisms divided into four parts: reasoning for perception, reasoning about object features, reasoning about the environment, and reasoning for planning. It also integrates a perception module to perceive the objects in the environment and specify the ontological knowledge [DADR19].
- The skill-based platform **SkiROS** combines low-level robot control and execution monitoring with an automated mission and task planning, and a high-level logistics planner, which includes a comprehensive World Model and communicates with a factory's manufacturing execution system. A key part of *SkiROS* is the World Model, which acts as a knowledge integration framework. The resulting system was successfully implemented in a simulated factory environment and was also tested in a real-world factory setting [CPT+16, RCH+17].
- The Agility Performance of Robotic Systems (**APRS**) project at the National Institute of Standards and Technology (NIST) developed an integrated agility framework, which enables manufacturers to assess and assure the agility performance of their robot systems [KKSG18]. The *APRS* project uses three ontologies (Workstation, Action, and Robot Capability), which are consistent with *CORA* (see Section 2.1) and applied to the kitting domain. The project infrastructure automatically transitions information from the ontology into the planning module based on PDDL and a PDDL executor subsequently. This executor then reads the PDDL actions as input and outputs a standardized set of low-level robot commands encoded in the Canonical Robotic Command Language (CRCL) [PBK+16].
- **ROSPlan** is a modular PDDL-task planning framework for the Robot Operating System (ROS) [CFL+15]. It provides ROS components (so-called ROS nodes) for

planning, problem generation, and plan execution, to enable the robot programmer easier integration of PDDL with existing data structures and algorithms from the ROS framework. Additionally, **OWL-ROSPlan** is an extension of ROSPlan for ontology support, using a specialized OWL-ontology as a knowledge base [BCM17].

2.4 Perception

Object recognition and localization play a key role in robotic systems, especially for autonomous robots, to implement bin-picking and assembly tasks. Methods that use two-dimensional images are widely used in the robotic domain for these purposes. In recent years, developed 3D vision technology and generated 3D point clouds provide better geometric, luminosity as well as depth information compared to 2D-vision and can be successfully used to recognize objects with less appearance information. There are two kinds of methods using global or local descriptors to recognize the object.

2.4.1 Local & Global Object Descriptors

The local descriptors are calculated for a single (key) point of the 3D object surface and can be used for object recognition through feature matching. This kind of descriptors is robust to occlusion and clutter but sensitive to the changes in the neighborhood around the points [HJX+18].

On the contrary, when using the global feature descriptor approach, a template library is generated by viewing the CAD model under different angles and computing a single global feature vector, which is later matched with one computed for the real object in the scene. The global feature descriptor reflects the characteristics of one view of the whole object. Moreover, compared with local methods, global methods require less memory, have a simpler and faster recognition procedure by computing fewer descriptors, which is highly important for applications running in nearly real-time [WLRC19].

Wohlkinger et al. presented a global shape descriptor that combines three shape functions (distance, angle, and area distributions on the surface of the partial point cloud) and classifies objects in real-time [WV11]. The Viewpoint Feature Histogram, which is composed of a viewpoint component and a surface shape component, is used for object recognition and pose estimation in a 6 degrees of freedom (6DOF) robot grasping operation [RBTH10]. Later, Wang et al. presented an improved feature descriptor named Orthogonal Viewpoint Feature Histogram (OVFH), where an improved viewpoint component is calculated using the orthogonal vector of the viewpoint direction [WLRC19]. A global texture-shape 3D feature descriptor is generated by extending the clustered viewpoint feature histogram with textured information. Further work relies on a global descriptor Globally Aligned Spatial Distribution (GASD).

2.4.2 Globally Aligned Spatial Distribution (GASD)

The global feature descriptor Globally Aligned Spatial Distribution (GASD) is based on the concepts of a global reference frame and globally aligned shape and color distributions [ST16]. The descriptor is computed using a partial viewpoint cloud of an object to derive the abstract descriptor vector from it. This procedure is can be done for multiple viewpoint clouds to estimate the best fitting viewpoint. Since viewpoints are known for training point clouds, this information can be used for rough camera viewpoint estimation to infer the pose of the object. The computation of the descriptor consists of two steps, the reference frame estimation and, the description of the shape with optional color information [ST16].

Reference frame estimation

The reference frame is estimated by a Principal Component Analysis approach. It uses the centroid of the point cloud as a reference frame center and the covariance matrix to derive the eigenvectors and eigenvalues. The eigenvector with the minimal associated eigenvalue is used as z-Axis of the reference frame and normalized in that manner to always point towards the viewer. Similarly, the eigenvector with the maximum eigenvalue is used as x-Axis of the reference frame. The y-Axis is completed as a constructed right-hand coordinate system based on the x and y-Axis. An example point cloud and the computed reference frame are visualized in Figure 2.6a Finally, all points of the point cloud are transformed into the reference frame's origin to describe the shape/color distribution [ST16].

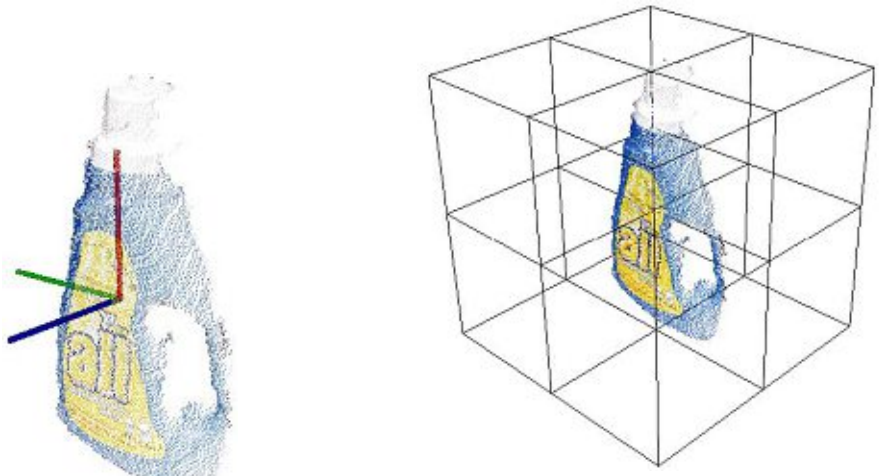
Description of shape/color

In the second step, the bounding box of the point cloud with the reference frame in the center is computed and uniformly subdivided into multiple grid cells. Such a subdivision into 2x2x2 cells is visualized in Figure 2.6b. For each grid cell, the number of points that lie within the cell is counted and stored. The grid cells are traversed to merge the resulting counts to a histogram. The histogram is normalized by the number of points in the point cloud [ST16].

Additionally, the color information of the point cloud can also be used for enhanced model description and therefore better matching results. To achieve better illumination invariance, the colors are transformed into Hue Saturation Value (HSV) color space. During the point counting procedure, an additional histogram of hue values per cell is accumulated. Similarly, these histograms are also normalized and concatenated to result in the final shape and color descriptor.

2.5 Summary

This chapter surveys previous work in the field of ontologies and planning with a special focus on their use in robotics. As stated in the introduction, there is a need for integrated



(a) The computed reference frame

(b) An example 2x2 grid for spatial distribution calculation

Figure 2.6: (a) The computed reference frame of GASD is shown in a right-hand coordinate system with x, y, z-Axis in red, green, blue, respectively. (b) A visualization of a 2x2 grid where the spatial distribution of points in each cell is computed. Images taken from [ST16].

frameworks which incorporate perception, automatic knowledge retrieval, planning and hardware execution. Some of the presented research (Section 2.2.3) only covers certain aspects of this need, but not as a connected framework. Nevertheless, there is still a significant shortcoming of the integrated knowledge frameworks (Section 2.3) concerning the domain/application specific tailoring. Therefore, changes in the application domain or the structure of the domain knowledge result in demanding changes in the knowledge retrieval and execution process.

CHAPTER 3

Knowledge-based Framework

In this thesis, an ontology-driven framework is presented to enhance the flexibility of industrial robotics systems and lower the required programming and configuration time for assembling new products in small batch size production. The core components of the knowledge-based framework are the World Model, Decision-Making, perception, and execution components responsible for controlling the robots. The components are integrated into a two-layer-based knowledge-intensive control architecture as presented in Figure 3.1.

The architecture is based on previous work, which separated the control of an individual manufacturing component into two levels: the High-Level Control (HLC) and Low-Level Control (LLC) [MHLL19, LZVM11]. A fundamental distinction between the layers is the ability of the LLC to execute in real-time, while the HLC might require longer computation time or even interactions with human operators. The HLC consists of the World Model and Decision-Making components. In this work, the previous architecture is extended in the framework with generic knowledge merging and mapping capabilities as well as with an integrated perception component, which is combined with robot manipulation capabilities. The presented framework is extensible and can automatically match the information from the perception system with the knowledge stored in the World Model for making conclusions about the current state. This state is then used for action planning to execute concrete commands for the robotics equipment.

3.1 World Model

The World Model is a central component of the framework with the role to connect all other linked components (perception, Decision-Making) and their subcomponents, which can read or provide information to it. The World Model aims to provide the actual and accurate model of the environment. For example, in the presented use-case, the World Model provides information about the main physical characteristics of the workpieces as

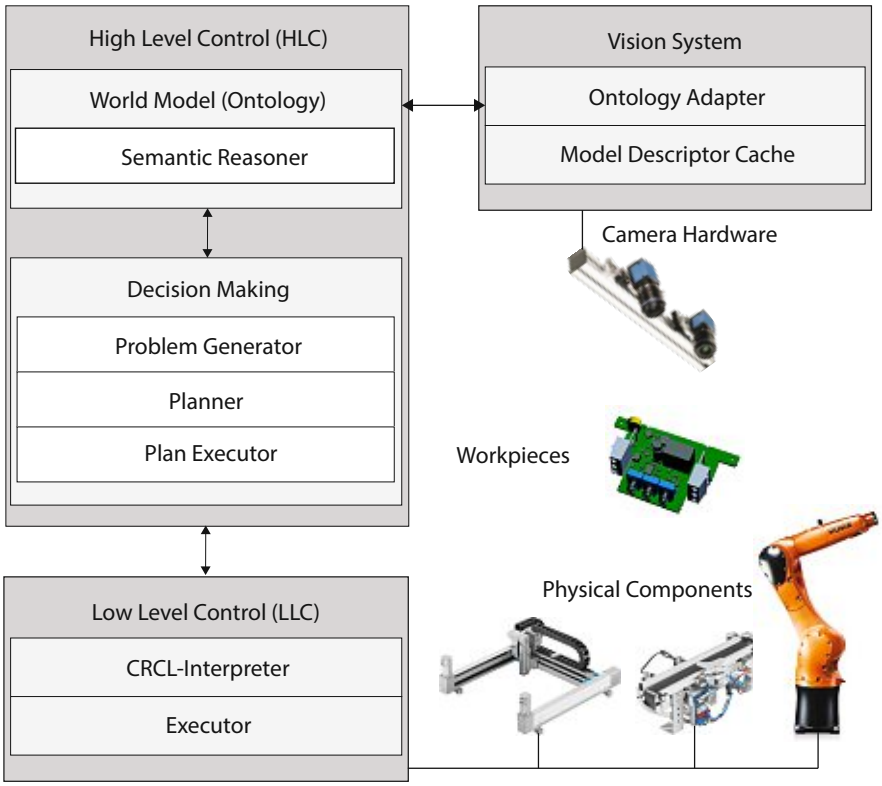


Figure 3.1: The core components of the knowledge-based framework, HLC, LLC, and Vision System, and their communication interfaces.

well as the relationships among its different features, such as the type of spatial relations, location, or orientation, which is of vital importance for automated path planning and scheduling.

It integrates three major services: knowledge acquisition, storage, and reasoning. To fulfill these demands, the World Model is stored in a heterogeneous ontology by incorporating the OWL. The knowledge acquisition is related to the process of populating the World Model and collecting information from sensor data. Furthermore, the World Model is used for storing the knowledge related to the domain of application, such as the semantic description of specific assembly operations, skills, assembly parts, and their features, tools, or states. The knowledge in the World Model can be easily extended with new information and concepts. Moreover, it can be linked with other existing ontologies to form a common structure for different applications.

3.1.1 Ontology Implementation

The implemented ontology is based on the ROSETTA ontology (see chapter 2.1.3) that focuses mostly on robotic devices and skills, with the skills being either primitive or

compound items for accomplishing some tasks within the manufacturing process [SM15]. This work strongly relies on the work of Perzylo et al. [PSRK15], OntoBREP, which presented a semantic description language for CAD models. This ontology stores knowledge about different CAD primitives and geometric constraints and establishes the link between product models and semantic robot task descriptions.

These works are extended with additional concepts related to assembly operations and constraints as well as with the perception module and the description of concepts needed to provide a complete CAD model of the used workpieces. The developed ontology focuses on concepts in the assembly automation domain and represents an accurate model of the robot environment including machines, tools, resources, skills and components, properties, operations, and services. The ontology framework is composed of six major concepts: Operation, Geometry, PhysicalObject, Property, Requirement, and Skill. The Property class in the ontology represents the characteristics of the objects, such as material, color, mass, robot, and gripper attributes (e.g. number and type of claws or joints) and sensor constraints (e.g. maximum and minimum range).

In assembly automation, some requirements have to be fulfilled to correctly manufacture the product, which is represented in the corresponding class. These requirements depend on various parameters such as the type of assembly operation, manufacturing parameters, and the material used as well as the workpiece's shape and dimensions. The PhysicalObject class represents objects that are tangible in contrast to the other classes represented in the ontology. The PhysicalObject includes the description of the robot and machine models, tools, and sensors as well as knowledge related to the products. A product is presented as a hierarchy of subassemblies and parts together with all their properties and relationships between them. Parts are defined as components, described by a set of attributes (geometry, color, etc.), properties, constraints, and relations to other parts. The connection to the product order is made later through the type of ordered product, quantity, which defines the number of parts that have to be available for starting the assembly process.

3.1.2 Semantic Reasoning

To understand its abilities and how to use them, a robot has to be able to reason about itself, the surrounding environment, possible actions, and their consequences [JMN16]. Symbolic reasoning is one of the main abilities that differentiate logic-based knowledge representation frameworks from other modeling approaches such as UML. It enables the system to check the consistency of the model and if a description is satisfactory. Symbolic reasoning is also used to infer additional information from the facts stated explicitly in the ontology [ZAF16, KSH14]. In this context, reasoning can be performed to infer object properties as well as relationships between individuals. The reasoning is used to infer if the robot can accomplish an action based on specific conditions that must be fulfilled or if tools can manipulate specific objects. Based on automated reasoning, the robot can also select the appropriate tool for an action to perform a specific operation. Besides, integrated reasoning makes robots more independent being able to understand

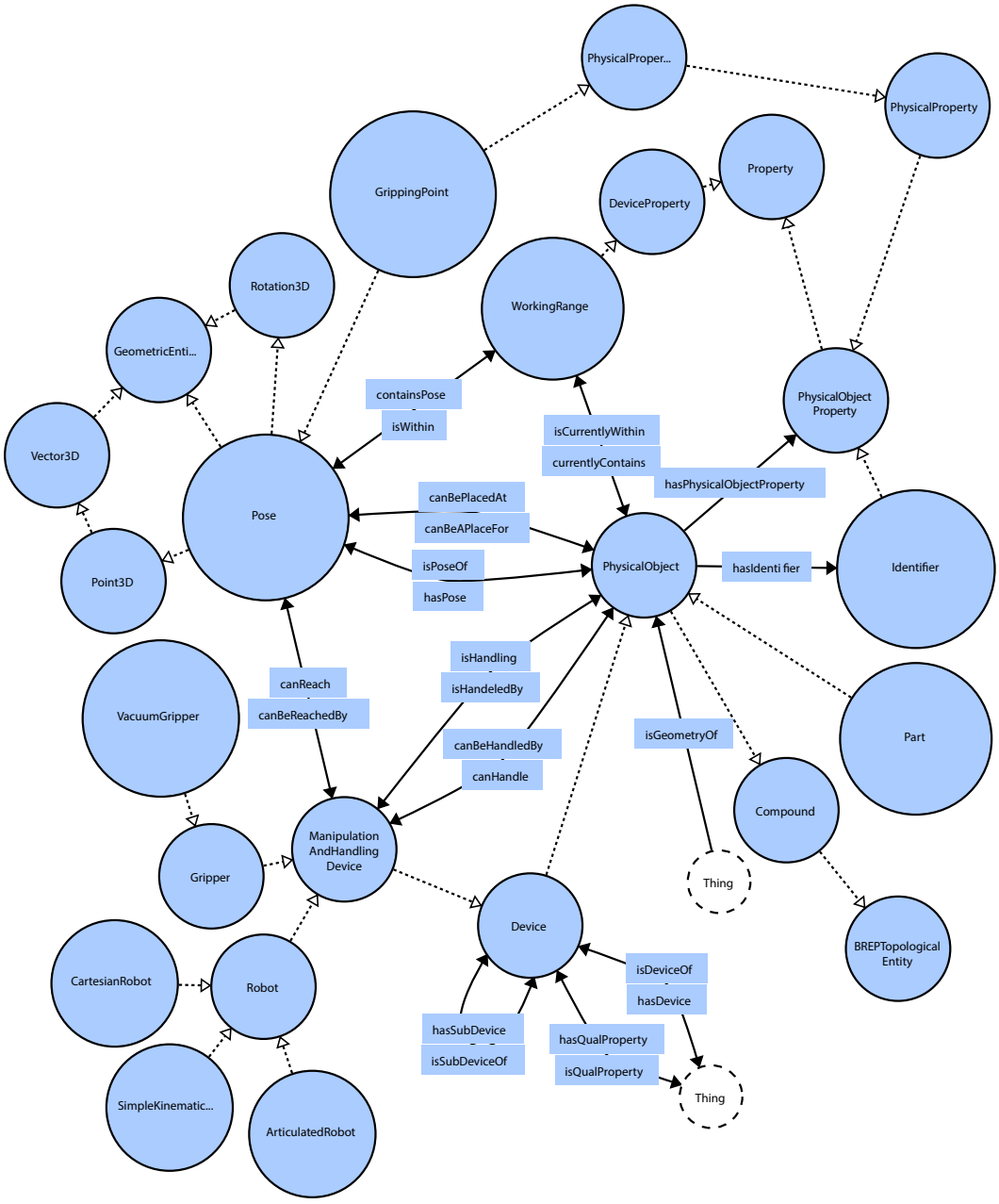


Figure 3.2: The important classes of the robotics pick-and-place World Model and their relations, visualized in the Visual Notation for OWL Ontologies (VOWL). The central classes are the *Pose*, *PhysicalObject*, *WorkingRange*, and the *ManipulationAndHandlingDevices* such as *Grippers* and three different kinds of robots. The central relations of this ontology are interconnecting these classes, such as *canReach*, *canBePlacedAt*, *isHandling*, *isWithin*, etc.

the actual manipulation constraints and conclude which position is reachable and how to grasp a workpiece.

The provided ontology can be used for automatic inference by a reasoner to deduce new knowledge from the existing one, but also to check the consistency and satisfiability of the provided semantic models. For instance, to accomplish pick-and-place operations, the robot has to reason about its capabilities to reach and grasp the different workpieces. In this context, the inference mechanism is used to deduce eligible grippers for handling particular workpieces (*Device*, *canHandle*). The inference mechanism can conclude, which robot can reach particular locations for grasping using the reachability relation (Position, canBeReachedBy). Moreover, semantic reasoning is used to analyze particular product parts as well as the relations between them, but also to investigate the feasibility of specific operations. In this context, reasoning about the conditions and constraints is applied to start specific actions based on reachability, grasping, and working areas. The most important classes and relations are shown in Figure 3.2 in the standardized Visual Notation for OWL Ontologies (VOWL) representation [LLMN15].

3.2 Decision-Making component

The Decision-Making component is responsible for generating and executing a plan that can bring the system from the deducted initial into the goal state. Generally, a plan is a sequence of granular state transitions such as picking or moving, which are called actions. In this case, the goal state is defined as the assembled final product.

The Decision-Making component is integrated with the World Model and the Low-Level Components to autonomously compute the actions while considering preconditions and other operational constraints. It consists of three consecutive main parts: the Problem Generator, the planner, and the Plan Executor. The Decision-Making component uses a planner implementing the PDDL [GKW+98] (see chapter 2.2.2) to try to find sequences of actions to achieve the goal conditions. To perform the planning, the Decision-Making component retrieves the initial world state as well as the goal state from the World Model to generate the PDDL description of the problem. For example, in the use-case, the generated PDDL-problem specification describes the initial state of the system and the instances of involved objects (cartesian robot, 6-axis robot, conveyor, workpieces, position, etc.) as well as the goal state to be reached.

3.2.1 Problem Generator

In order to provide a more general architecture suitable for automating planning, a generic PDDL Generator is introduced. The role of this PDDL Generator is to automatically query the required information from the World Model for the planner. The generator translates concepts provided in the World Model into data structures of the PDDL-problem and domain. The mapping scheme to translate from OWL to PDDL is described in previous work [HLM19], as shown in Figure 3.4.

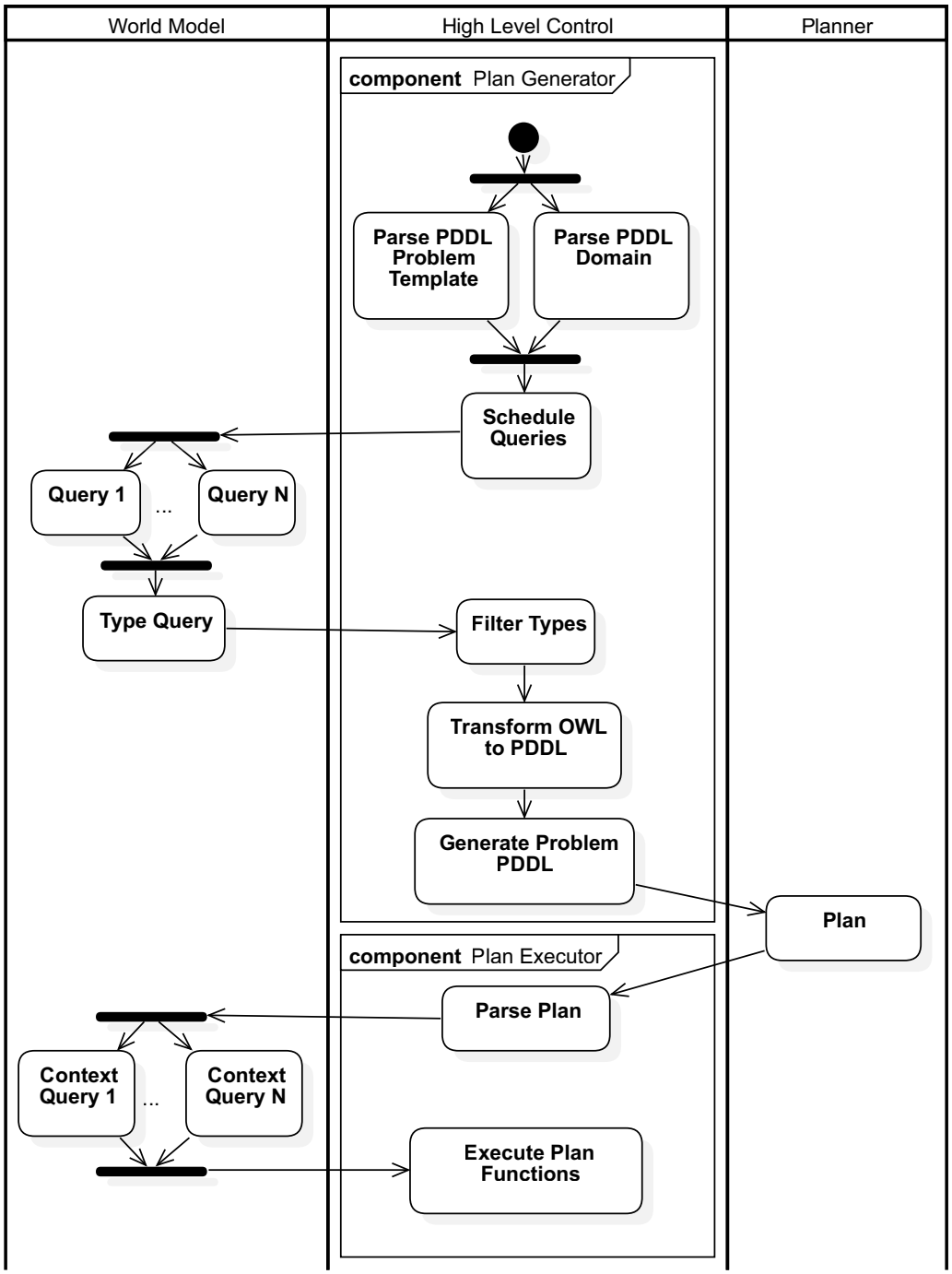


Figure 3.3: The processing pipeline of the Decision-Making component with its three parts: problem generation, planning, and plan execution.

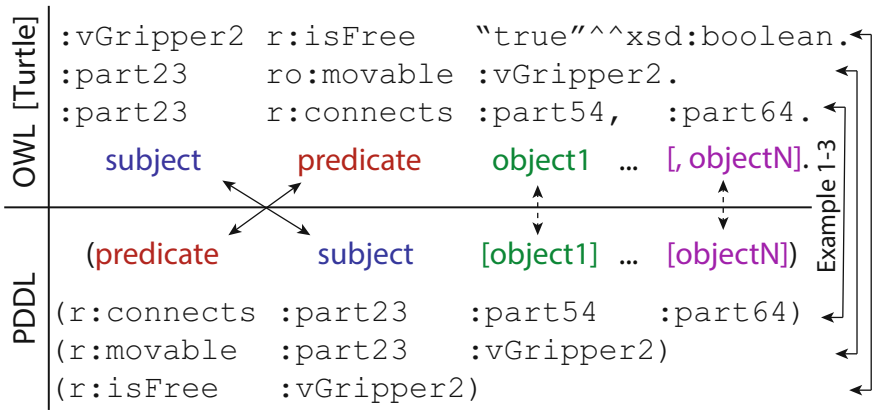


Figure 3.4: OWL-PDDL-Mapping scheme: Three examples with different parameter lengths are listed to illustrate the bi-directional mapping between OWL triples and PDDL predicates [HLM19]. The OWL-triples are listed in Turtle syntax, with two predefined namespaces: 'r' and the empty prefix. Image take from previous work [HLM19].

The World Model uses OWL-triples/quads as data description to relate a subject with a predicate to an object (and the context as quad). The data structure of PDDL is focused on predicates that relate a predicate name with multiple parameters. Depending on the number of parameters, these PDDL-predicates and OWL-triples can be mapped by relating the first two PDDL-predicate parameters with the subject and object of the OWL-triple, as shown in Figure 3.4. Missing parameters can be automatically completed, but excessive PDDL-predicates with more than two parameters can only be expressed with multiple OWL-triples, therefore should be modeled differently. The execution pipeline of the PDDL Generator is shown in Figure 3.3.

PDDL generation

In the first place, the generator parses the user-specified domain to retrieve the domain-specific types, predicates, actions, etc. In the following step, the generator uses a problem template that specifies the required information (objects, initial state, goal) from the World Model. This can be a generic problem template as specified in Listing 3.1, or can be extended with static domain-specific information that is not stored in the World Model. Multiple commands are implemented in the template engine to query either specific predicates, all predicates from the domain, or specific results from a defined SPARQL-query[1]. These commands are also extendable with a filter for only receiving information from specified sub-graphs. Additionally, the types of all received objects are automatically queried and listed with their names for the PDDL-object-type definitions. Even though the colon (":") is not a valid symbol for PDDL-predicate names, it is

[1] https://www.w3.org/TR/rdf-sparql-query/

```
1  (define (problem Robot-PickPlace)
2      (:domain Robot)
3      (:objects
4          {{individualTypes}}
5      )
6      (:init
7          {{predicates graph=":ontology"}}
8      )
9      (:goal (and
10         {{predicates graph=":goal"}}
11     ))
12 )
```

Listing 3.1: PDDL-problem template with three template commands in curly brackets. The first command is replaced with all queried named individuals and their types. The second and third command is replaced with all triples from the specified subgraph where the predicate name is mentioned in the PDDL-domain.

tolerated by planner implementations, for example, Fast-Downward[2]. To overcome this syntactical issue, this RDF prefix symbol can be substituted with a defined unique symbol sequence, for example, underline, dash, underline ("_-_"), by the PDDL-generator before planning and substituted back after planning.

In the next step, the generator passes this information to the planner, which searches for a solution for the generated problem.

3.2.2 Plan Executor

The resulting plan consists of a list of PDDL-actions with parameters to get to the specified goal. This resulting plan is parsed by the Plan Executor to perform these actions. For this purpose, additional context information is needed. For example, the action to pick up a workpiece needs additional information about where the workpiece is located and how to pick it up. For that purpose, the implemented framework enables the specification of context queries that are executed for each instance of a specified type if this instance is mentioned as a PDDL-action parameter in the generated plan. In other words, for each instance that is mentioned in the plan and therefore represented in the ontology, a query can be executed automatically. For example, for each workpiece that is considered in the plan, the gripping position is automatically retrieved. With that additional information, the actions can be executed directly. For this purpose, concrete commands in the CRCL syntax are sent to the Low-Level Control, which executes them on the physical hardware.

Besides the specification of the OWL-ontology and its related PDDL-domain, the context queries and the execution of the plan actions are the only components that need to be coded for the domain-specific use case. The whole ontology querying and PDDL-generation is generically implemented to enable a use-case independent application with

[2]Fast-Downward PDDL-Planner, available at `http://www.fast-downward.org/`

minimal implementation effort. This framework not only simplifies the implementation of domain-specific plan execution but completely removes the need to implement data querying and PDDL-generation.

Decision-Making component implementation

Because of the processed semantic web data, the Decision-Making component is also implemented using web technology, especially Javascript in the Node.JS[3] runtime environment. For this purpose, the framework is built on RDF/JS[4], N3.js[5] and SPARQL.js[6]. Additionally, the framework uses Parsimmon[7] and Handlebars.js[8] for PDDL-parsing and PDDL-generation, respectively. Finally, all 3D-processing is done using Babylon.js[9]

3.2.3 Decision-Making Component Evaluation

The Decision-Making component is evaluated using a test set based on 25 PDDL instances from previous International Planning Competition (IPC)[10] of the years 1998 to 2018. These 25 PDDL instances are manually converted to OWL-triples to show the correct query and PDDL generation by comparing the generated PDDL instance with the original PDDL format. This should also evaluate the performance and generalization possibilities of the system, to enable the usage of the system in a broad application range, even outside of the robotics domain.

Evaluation test set Based on 11784 PDDL-problems from all competitions, a random subset is selected as listed in Table 3.1. For that subset, a maximum of 3 test-instances is selected for each competition year, since the number of test-instances varies highly from year to year. If a competition instance contains multiple domains/problems, one random problem/domain pair is selected. Additionally, all test-instances with an excessive number of predicate parameters are filtered, as mentioned before. Additionally, the current implementation does not support the full PDDL standard. Therefore, disjunctive (or-associated) goals as well as specially defined metrics are not supported and excluded from the test set.

For each evaluation instance, the following metrics are evaluated: the execution duration, the file size of the original PDDL-problem, the file size and the number of converted quads. The execution duration should only be linearly proportional to the problem-size, to apply this implementation practically. As listed in Table 3.1, the execution times are in the same order of magnitude compared to the problem sizes.

[3] NodeJS, available at `http://nodejs.org/`
[4] RDF/JS Data model specification, available at `https://rdf.js.org/data-model-spec/`
[5] N3.js, available at `https://github.com/rdfjs/N3.js/`
[6] SPARQL.js, available at `https://github.com/RubenVerborgh/SPARQL.js/`
[7] Parsimmon, available at `https://github.com/jneen/parsimmon`
[8] Handlebars.js, available at `https://handlebarsjs.com/`
[9] Babylon.js, available at `https://www.babylonjs.com/`
[10] `https://www.icaps-conference.org/competitions/`

Problem	Quads	Quads [kB]	PDDL [kB]	Duration [ms]
ipc-2004-psr-small-strips-d31-p31	28	2.1	0.5	27.2
ipc-2004-satellite-strips-p11	29	2.0	1.0	26.9
ipc-2000-blocks-strips-typed-p27	50	2.2	0.5	17.4
ipc-2000-blocks-strips-untyped-p49	54	2.7	1.0	22.2
ipc-1998-gripper-round-1-strips-p18	61	3.3	1.9	29.2
ipc-2004-airport-nontemporal-strips-d22-p22	61	3.0	3.1	30.2
ipc-2002-driverlog-strips-automatic-p18	64	2.4	1.1	22.4
ipc-2018-caldera-opt-temporal-p2	85	4.3	3.3	141.0
ipc-2014-barman-sequential-optimal-p2	89	4.0	2.0	46.6
ipc-2006-openstacks-propositional-p27	93	3.4	1.5	29.3
ipc-2002-freecell-strips-automatic-p13	130	4.2	2.6	27.2
ipc-2016-bottleneck-p6	132	5.2	4.0	110.7
ipc-2014-barman-sequential-multi-core-p9	134	5.6	3.0	114.6
ipc-2014-barman-sequential-satisficing-p2	137	5.8	3.0	71.3
ipc-1998-mystery-round-1-strips-p26	159	6.7	5.6	30.0
ipc-2016-bottleneck-merge-and-shrink-set-p13	240	8.5	7.2	139.0
ipc-2000-freecell-strips-untyped-p51	291	8.3	6.0	39.1
ipc-1998-grid-round-2-strips-p2	320	10.4	10.3	101.9
ipc-2002-depots-strips-hand-coded-p10	592	18.0	10.8	62.2
ipc-2018-organic-synthesis-sat-temporal-d18-p18	594	12.4	12.6	76.2
ipc-2011-visit-all-sequential-optimal-p10	684	18.3	19.2	78.3
ipc-2018-nurikabe-opt-temporal-p16	746	17.4	19.3	190.1
ipc-2011-visit-all-sequential-satisficing-p11	1122	29.6	32.2	68.9
ipc-2011-visit-all-sequential-multi-core-p13	1474	38.7	42.8	103.2
ipc-2016-chessboard-pebbling-p20	2453	75.9	77.3	174.4

Table 3.1: Evaluation of the Decision-Making-Component by measuring the runtime of 25 PDDL instances from previous International Planning Competitions competitions.

3.3 Perception component

To flexibly act in a dynamic environment, robots need to combine the perception abilities and reasoning skills to understand the captured images. The focus of the perception component in the framework is to localize and identify targeted workpieces. It is linked to the Decision-Making component by integrating the perception algorithms with the World Model with the help of an Ontology Adapter and a Model Descriptor Cache. As presented in Figure 3.5, the recognition process consists of two main steps: offline and online recognition.

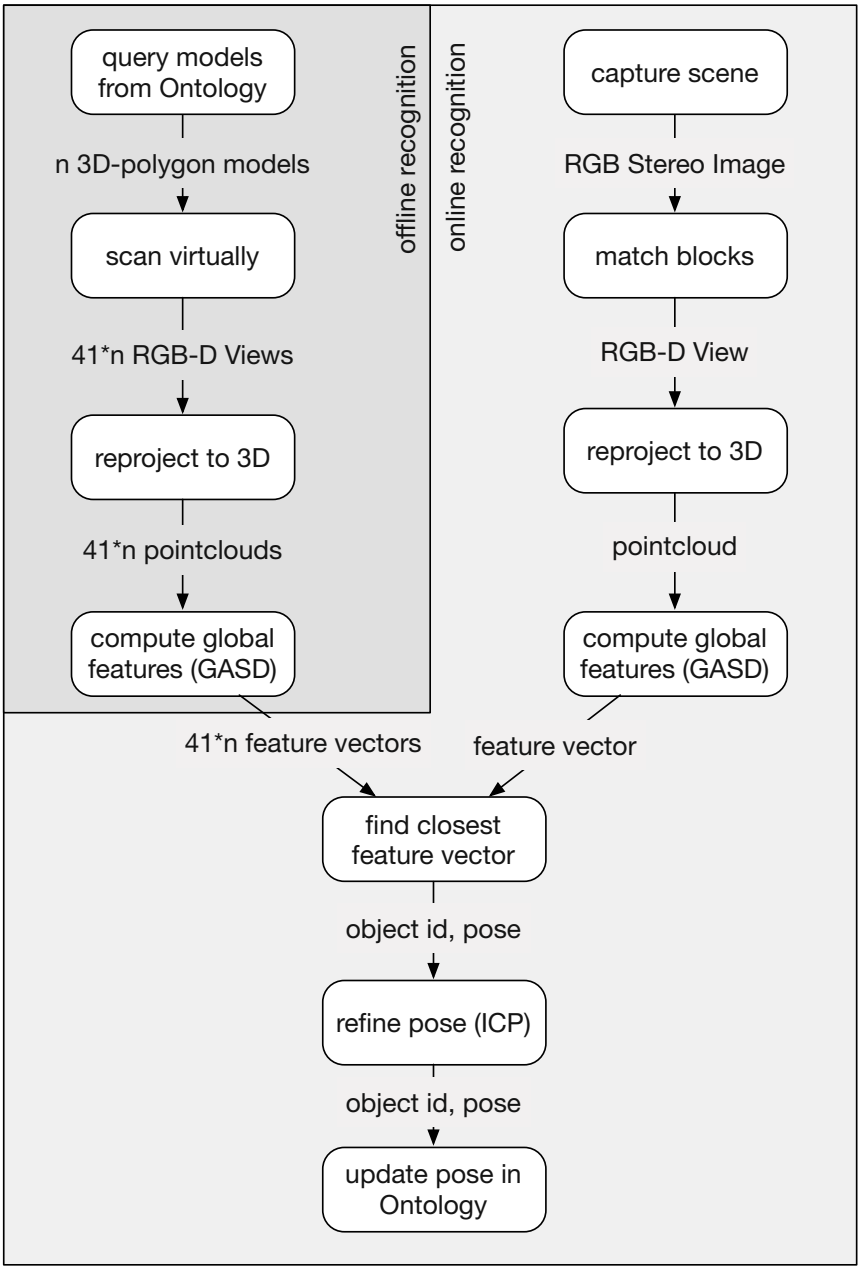


Figure 3.5: The object recognition process with the two distinct processing phases: offline and online.

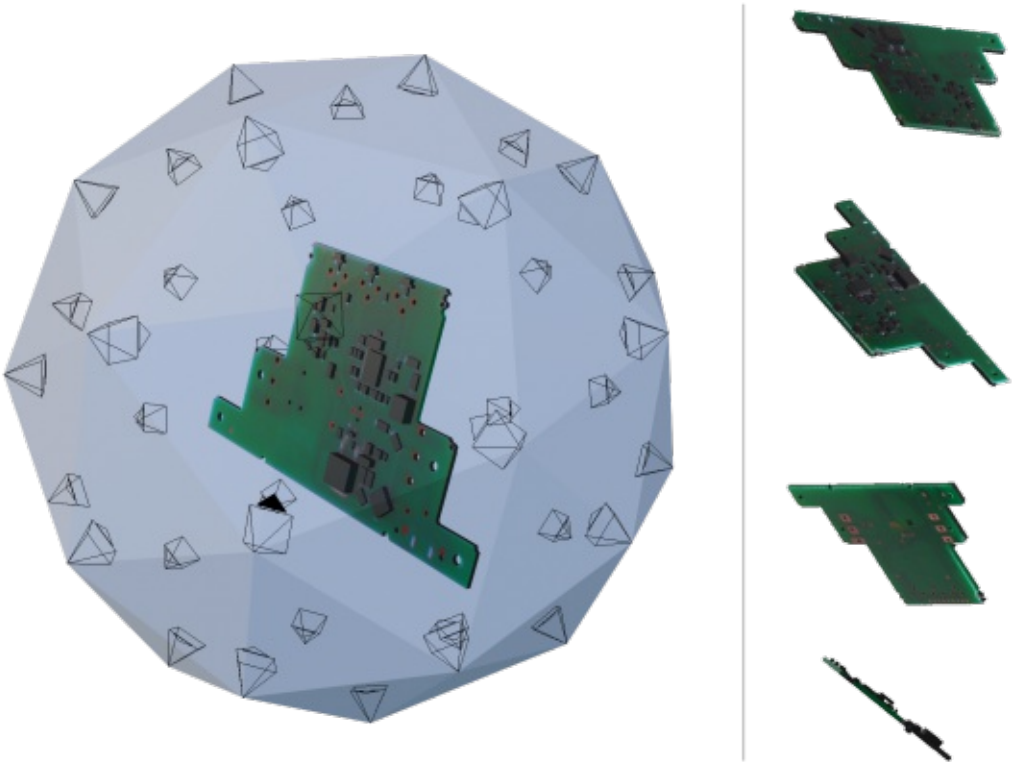


Figure 3.6: The virtual scanning environment with 41 cameras placed on the corners of a polyhedron around the scanned workpiece, which is a printed circuit board in the given case. Four perspectives are shown for illustration on the right.

3.3.1 Offline Recognition

The offline recognition step is designated to capture the template objects' CAD models from different viewing angles to populate the Model Descriptor Cache with global feature descriptors of them. The CAD models are stored in the ontology using the OntoBREP semantics (see chapter 2.1.3), which describes the relations between an object's faces, edges, and vertices. The CAD models are queried from the ontology by the Ontology Adapter and virtually scanned as Red Green Blue - Depth (RGB-D) images (three color channels with an additional depth channel) by 41 radially evenly distributed cameras, as shown in Figure 3.6. By using the perspective transformation matrix from the virtual stereo camera, each RGB-D image can be reprojected to a 3D point cloud [Sze10]. For each 3D point cloud, a global feature descriptor is computed and stored in the Model Descriptor Cache to find the most similar object in the online step.

The presented approach relies on the research presented by Lima et al. [ST16] (see chapter 2.4.2). This proposed global point cloud description method takes as input a 3D point cloud that represents a partial view of a given object. Firstly, a reference frame for the

point cloud is estimated. Afterward, a shape descriptor for the point cloud based on the spatial distribution of the 3D points is calculated. The color distribution is also considered to obtain a combined shape and color descriptor with a higher discriminative power.

3.3.2 Online Recognition

In the online step, similar processing steps are applied to compute a global feature descriptor from the data of the stereo camera. In detail, a disparity image is generated from the two Red Green Blue (RGB) images of a passive stereo camera with a block matching scheme. After reprojection to 3D, the background of the point cloud is filtered and clustered to segment the recognition of candidate objects. For each recognition candidate, the global feature descriptor is computed to find the most similar template object in the Model Descriptor Cache. This search is performed with the L2-norm on the global feature GASD-descriptor. The recognition outcome is the template, which has the most similar feature descriptor histogram with the point cloud of the real object. Finally, the approximate orientation of the object is refined using the Iterative Closest Point (ICP) algorithm [CM92]. The resulting pose as a combination of position and orientation is updated in the World Model by the Ontology Adapter.

3.3.3 Vision System Implementation

Since the stereo block-matching algorithm for point-cloud reconstruction relies on a variety of parameters (disparity range, block size, filter values, etc.), an intuitive user interface is implemented as shown in Figure 3.7. By providing live 3D point cloud feedback, especially small artifacts around object edges can be observed and avoided with different parameters that usually are occluded in the depth image view.

The object recognition pipeline is implemented using the Point Cloud Library (PCL) [RC11], whereas the User interface relies on Qt[11] OpenCV[12], Compute Unified Device Architecture (CUDA)[13], the StereoVision library[14] and VisPy[15].

In application, different objects involved in the assembly process such as Printed circuit board (PCB) as well as different Through-hole technology (THT) devices (relays, capacitors, and potentiometers) are used, with the smallest object measuring only 7 mm in length. In contrast to active RGB-D sensors such as the Microsoft Kinect V1/V2[16] and the Intel Realsense R435[17], both previously used in past work, the passive stereo vision sensor is not limited by a minimal depth constraint which restricts the spatial resolution. On the contrary, this passive sensor in combination with block-matching can

[11] Qt, available at `https://www.qt.io/`
[12] OpenCV, available at `https://opencv.org/`
[13] CUDA, available at `https://developer.nvidia.com/cuda-downloads`
[14] StereoVision library, available at `https://github.com/erget/StereoVision`
[15] VisPy, available at `https://github.com/vispy/vispy`
[16] `https://developer.microsoft.com/de-de/windows/kinect/`
[17] `https://www.intelrealsense.com/depth-camera-d435/`

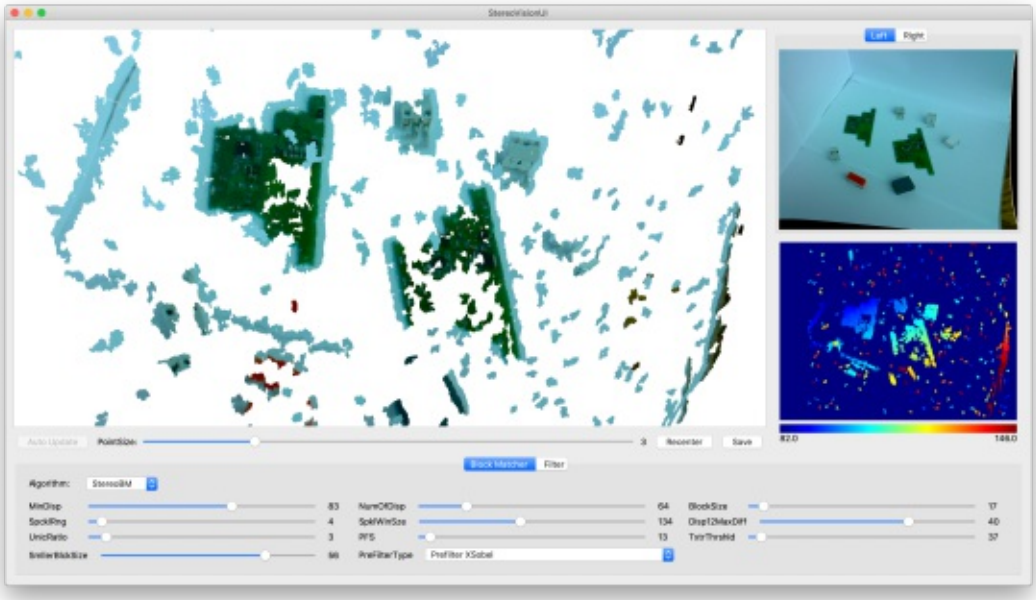


Figure 3.7: Screenshot of the user interface to calibrate the parameters of the stereo block-matching algorithm. Either the left or right RGB camera image can be shown in the upper right, which is matched with the configured parameters of the bottom to result in a depth image visible on the right. By using the camera calibration data, a point cloud viewer shows the resulting data in 3D in the main view.

only provide depth information for color/intensity-changes around edges and corners of the object but not for textureless objects.

Another advantage observed in the experiments with the global descriptors is the higher invariance to noise in contrast to a local-descriptor-based approach such as Fast Point Feature Histograms (FPFH) [RBB09]. In the case of the object being captured with low spatial resolution or high spatial noise, the correspondence grouping algorithm can not identify any matches, but the global-matching approach yields a correct match (with a relatively high distance).

In sum, sufficient recognition results for this use-case are achieved in metrics of precision for the PCBs and accuracy due to the well-provided spatial resolution. However, with the relatively inexpensive consumer camera and lens setup, this approach was not able to identify smaller objects (less than 30 mm) without additional optics equipment.

3.4 Execution component

The purpose of the Low-Level control is to integrate the planning system with the execution component for carrying out the abstract CRCL commands on the physical

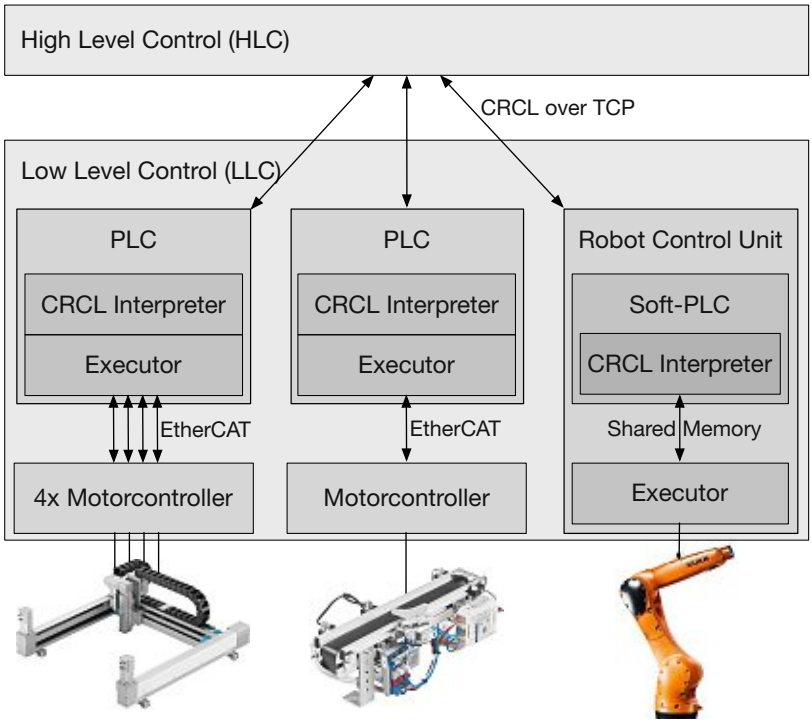


Figure 3.8: Overview of the low-level components and their communication channels.

components.

The CRCL generator of the Decision-Making component sends the planned commands over Transmission Control Protocol (TCP) to a CRCL interpreter of the low-level component and monitors each command's status reply. A generic CRCL interpreter is implemented as a standardized IEC-61131 (International Electrotechnical Commission) program to run on a Programmable logic controller (PLC) or a SoftPLC (Software PLC). The purpose of this CRCL interpreter is the parsing of CRCL commands. For each PLC, only the hardware-specific execution has to be implemented. In the presented use case, a conveyor and two different robots are used: an articulated 6-axis Kuka robot as well as a Festo cartesian robot, as presented in Figure 3.8.

The CRCL interpreter parses the CRCL commands and notifies the sender when the execution of the command starts and finishes. To support the web-based application, the standard XML-based syntax of the CRCL is replaced by JSON. This JSON format is also used since it has a lower message size. Currently, the CRCL client sends CRCL commands to the components PLCs via TCP-sockets but classical representational state transfer (REST) interfaces are possible as well.

CHAPTER 4

Application

4.1 Scenario

The application focuses on the efficient automation of an assembly process that covers the mounting of different types of electronic devices for a price-sensitive product series up to a lot size of 5. This includes the assembly of different THT devices on a PCB as shown in Figure 4.1. In this context, the involved manufacturing components have to manipulate (pick-and-place) related workpieces (the THT devices) as well as to transport (pass) them between locations to assembly them in the final configuration. The assembly of THT devices is harder than the assembly of Surface-Mounted Devices (SMDs) since the pins of THT devices are not uniform, i.e. they can be deformed. This is also a reason why this process is currently performed manually by human workers.

The application consists of two robots, an articulated 6-axis Kuka robot and a Festo cartesian robot, as well as a conveyor belt, which serves as a transportation unit that

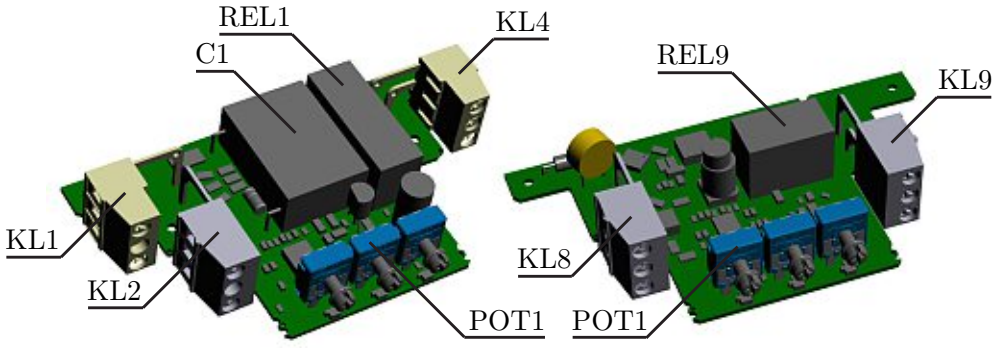


Figure 4.1: Two variants of PCB assembly configurations which define the type and pose of the nine different THT devices, labeled by name.

Figure 4.2: In the left image, the application environment is shown, which consists of a cartesian Festo robot, a connecting conveyor belt, and an articulated 6-axis robot by Kuka (from left to right). In the right image, the assembled PCB is shown next to the stereo sensor.

moves the THT devices between them (see Figure 4.2). The Kuka robot is equipped with two vacuum grippers to transport two parts simultaneously. The conveyor belt can transport a pallet with eight fixtures, where each fixture is capable to hold different THT-devices with a pneumatic cylinder mechanism.

To test the system's flexibility of the assembly line in different scenarios, two variants of PCB assembly configurations are focused which define the type and pose of the assembled THT devices (see Figure 4.1). Therefore, the application also involves object localization and recognition skills: A stereo camera can detect the type and pose of the PCB to determine the PCB configuration. Based on the PCB configuration, the necessary THT devices can be picked from storage trays and placed on the conveyor pallet fixture by the cartesian robot, transported to the articulated robot with the conveyor, and finally assembled on the previously localized PCB.

The robots are equipped with vacuum grippers with different nozzle sizes to handle the THT devices. Because of the different physical properties of the THT devices, each THT device can only be handled by certain types of grippers, e.g. the bigger relays can only be picked by the vacuum gripper with the bigger 4mm nozzle, but not the smaller 2mm nozzle. In many cases, a THT device can be handled with multiple grippers, which helps to decrease assembly time by picking multiple THT devices at the same time before moving to another working area.

4.2 Use-Case Procedure

In the use case, once the PCB is identified by the vision system which saved this information in the ontology, the Decision-Making component uses the ontology model to plan the necessary assembly operations (e.g. picking from trays, passing between robots

```
1 SELECT DISTINCT *
2 WHERE {
3   GRAPH :goal {
4     ?s ?p ?o .
5     FILTER (?p IN (ro:isSubDeviceOf, :canHandle,
6       :hasPose, :isCurrentlyWithin, :isHandling,
7       :canReach, :containsPose, :canBePlacedAt)).
8   }
9 }
```

Listing 4.1: The generated SPARQL query queries all entities and relations from the goal-subgraph.

and conveyor pallet fixtures, moving the robots Tool Center Point (TCP) or the conveyor pallet).

The objective of the Decision-Making component is to automatically generate and execute coherent plans for each physical device (i.e. robot, conveyor) based on the current state of the World Model and the required skill for each operation in the assembly process so that the product can be produced. For this purpose, the entire assembly environment is represented in the World Model including the type, position, tools, and skills of the robots, and information about the conveyor and of the pallet that is composed of multiple working areas as well as the model and position of the THT devices in the trays. These entities are related by several associated properties, such as *canBeReachedBy*, *canHandle*, *isHandling*, etc., as visualized in Figure 3.2

4.2.1 Problem Generation

As mentioned in section 3.2, these entities and relations are passed to the planner in the form of a PDDL-problem as specified in the PDDL-problem template in Listing 3.1. Based on this template, SPARQL-queries are generated by the PDDL-generator for template completion. For example, the generated SPARQL-query to retrieve all entities and relations from the goal-subgraph which are specified in the PDDL-domain is shown in Listing 4.1. This query yields not only all explicit information from the ontology to the planner, but also all reasoned implicit information.

4.2.2 Planning

After completing the PDDL-generation, the PDDL-planner tries to produce an optimal plan which satisfies the goal, as mentioned in section 3.2. In the use-case, several preconditions have to be fulfilled for the actions as stated in the pick-action in Listing 4.2. The robot and the THT device to be picked up have to be within the same working range (*:hasPose*, *:canReach* and *isCurrentlyWithin*) which need to be reachable (*canReach*). Additionally, the gripper must be able to handle the part (*:canHandle*) and has to be attached to the robot (*ro:isSubDeviceOf*). Additionally, the gripper needs to be free

```
(:action pick
 :parameters (?robot - ro:Robot
  ?gripper - ro:Gripper ?part - ro:PhysicalObject
  ?pose - cad:Pose ?range - ro:WorkingRange)
 :precondition (and (:hasPose ?part ?pose)
  (:canReach ?robot ?pose)
  (:isCurrentlyWithin ?robot ?range)
  (ro:isSubDeviceOf ?gripper ?robot)
  (:canHandle ?gripper ?part)
  (not (isHandlingPart ?gripper))
  (:containsPose ?range ?pose)
 )
 :effect (and (:isHandling ?gripper ?part)
  (isHandlingPart ?gripper)
  (not (:hasPose ?part ?pose))
 )
)
```

Listing 4.2: The picking action of the PDDL-domain in the use-case.

(*isHandlingPart*); in other words, no other object should be already handled with the gripper.

To handle the move of a THT device from one robot to another, a similar passing action is implemented which unifies the place action of the first robot and the pick action of the second robot in one action.

Additionally, the system reasons about the optimal tool usage and conveyor transport pallet fixture placement. In the current configuration, the ontology contains information about which THT device can be handled with which vacuum gripper and which pallet fixture. This information is used by the planner to calculate a plan that includes the gripping of multiple parts before moving between working areas to minimize the overall execution time by minimizing the number of moves. This is also applied to the conveyor: since each side of the conveyor is within a different working area, the planner tries to minimize the conveyor moves by maximizing the transport pallet utilization.

4.2.3 Planner Evaluation

For this three robots assembly use-case, the generated PDDL-problem consists of 46 objects and 112 predicate definitions, which are queried from the ontology and converted into PDDL within 387ms. Multiple PDDL planners are integrated into the framework to evaluate and use the planner with the best performance. As shown in Figure 4.3, 15 different planning algorithm configurations are utilized to measure calculation time and plan costs. In the use case, the plan costs are equivalent to the number of actions to reach the goal. Each planning evaluation is limited to 60 seconds. Planning with the fastest planning algorithm configuration (Fast-Downward with Lama first) took 2.1 seconds for a solution with 37 actions. The generated plan from this planner is shown in Listing 4.3.

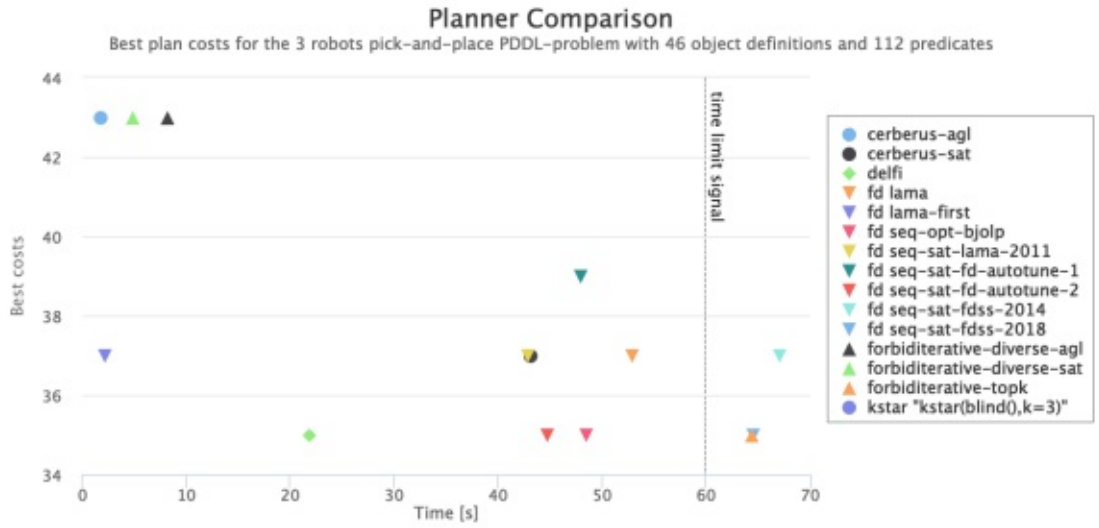


Figure 4.3: Performance evaluation of 15 different planning algorithms.

```
1  (pick :robot_festo :vacuumgripper_festo_2mm :part_kl4 :pose_kl4
       :workingrange_init)
2  (move :robot_festo :workingrange_init :workingrange_conveyor_festo)
3  (pass :robot_festo :robot_conveyor :vacuumgripper_festo_2mm
       :vacuumgripper_conveyor_4 :part_kl4 :pose_conveyor_festo
       :workingrange_conveyor_festo)
4  (move :robot_festo :workingrange_conveyor_festo :workingrange_init)
5  (pick :robot_festo :vacuumgripper_festo_2mm :part_rel9 :pose_rel9
       :workingrange_init)
6  (move :robot_festo :workingrange_init :workingrange_conveyor_festo)
7  (pass :robot_festo :robot_conveyor :vacuumgripper_festo_2mm
       :vacuumgripper_conveyor_5 :part_rel9 :pose_conveyor_festo
       :workingrange_conveyor_festo)
8  ...
9  (pass :robot_conveyor :robot_kuka :vacuumgripper_conveyor_2
       :vacuumgripper_kuka_2mm :part_kl1 :pose_conveyor_kuka
       :workingrange_conveyor_kuka)
10 (pass :robot_conveyor :robot_kuka :vacuumgripper_conveyor_1
       :vacuumgripper_kuka_4mm :part_kl2 :pose_conveyor_kuka
       :workingrange_conveyor_kuka)
11 (move :robot_kuka :workingrange_conveyor_kuka :workingrange_goal)
12 (drop :robot_kuka :vacuumgripper_kuka_2mm :part_kl1 :pose_kl1_goal
       :workingrange_goal)
13 (drop :robot_kuka :vacuumgripper_kuka_4mm :part_kl2 :pose_kl2_goal
       :workingrange_goal)
```

Listing 4.3: The generated plan from the planner for the three robot assembly tasks. The first 7 actions show the picking of the assembly parts with the Festo robot to pass them to the conveyor, before the conveyor moves. The last 5 actions show the multi gripper behaviour to handle two parts simultaneously before moving to the next working area to save time

4.2.4 Plan Execution

As mentioned previously, after planning and plan parsing, the plan executor needs to query additional context information from the ontology for the execution of the abstract plan actions. For all these actions, which operate with positions, the abstract positions are retrieved from the World Model with the concrete coordinates by the context queries. The exact positions are retrieved by combining the planned position with the picking point of the object. Since the positions are also specified by a three-dimensional rotation, the so-called poses are combined by matrix multiplication of homogeneous coordinates. Depending on the motion planning capabilities of the robot, the additional combination of this resulting pose with the current TCP is done on the low-level component.

Finally, the list of actions is sent to the relevant component together with the information necessary for executing the task.

CHAPTER 5

Discussion

The motivation of the presented and applied knowledge-based framework is to provide more autonomy for robotics systems. The main focus is to enable the robot to understand the actual environmental conditions so that it is able to plan and act according to the current situation. In this context, the developed framework includes perception, reasoning, and planning abilities, which support the ability of the robotics system to adapt to changing demands and increasing product variants, which require agile action and flexibility.

The ontology is not only used to represent the products and parts but also to describe the types of available resources and their relationships within the production environment. This available knowledge supports the system flexibility regarding object detection and localization as well as various manipulation actions. Within the framework, the developed knowledge base provides a comprehensive source of information, which is necessary for successful robot task execution. We integrated different types of robots that receive the queried knowledge from the knowledge base to be able to handle different manipulation actions for proving the flexibility of the approach. Consequently, this generic approach is applicable for assembling different objects with different types of robots.

Integrated reasoning makes robots more independent by being able to understand the actual manipulation constraints and conclude which position is reachable and how to grasp a workpiece. The presented approach shows, without concrete quantification, the significant potential to reduce programming efforts, as the presented planning approach automatically generates the commands to be executed by the low-level component. Besides, the defined action models are generic and separated from the concrete problem description. This makes them more reusable for creating code for different types of robots and product configurations.

The action models are then continuously matched with the problem models that represent the application domain as well as the current real-world state, which is subject to changes

and continuously adapted. This also facilitates easier uncoupling from the underlying execution system as the same actions can be executed with different robot types if an appropriate interface is provided. We proved in our application the generalization capabilities of the framework by interconnecting two different types of robots from two different manufacturers. Moreover, the level of required robotic knowledge is considerably lowered, considering that the user has to specify the product configuration and workpiece description on an abstract level. In other words, the knowledge needed for robotic application implementation is moved from robotics experts into the framework, and domain experts can deploy specific assembly or manufacturing tasks easier. This removes the need for manually programming each robot activity, whenever a new product is introduced. The use-case is designed and deployed to show the applicability of the knowledge-based approach for handling the assembly of small lot size products.

One of the shortcomings of our approach is that the gained flexibility is related to the scale and quality of the content provided in the knowledge base, as all states and effects that influence the robot's actions have to be explicitly represented in the word model. Otherwise, a plan may not be executable if the description of the current situation is not complete or inaccurate. As a result, the modeling of the systems objects, states, relationships, constraints, and actions can be a very extensive process for some application domains. Further shortcomings are related to the employed inexpensive perception equipment as the current setup does not provide the required minimal spatial resolution to recognize small parts.

CHAPTER 6

Conclusion

To handle the ever-increasing number of individualized products with short delivery times, production systems have to become more flexible and agile. Robotics technology, which can provide high efficiency, precision, and repeatability, is regarded as a viable solution to cope with these challenges. However, robotics systems still do not meet completely the demands of small- and medium-sized enterprises (SMEs) and significant time-consuming efforts are still needed to realize a fully automated production of small lot sizes. Typical industrial robotics systems are not flexible enough and new methods, which are able to autonomously generate and execute process plans, should be introduced to respond to rapidly changing demands in the production domain.

It is anticipated that increasing the robots' knowledge and autonomy could lead to more agile and flexible robotics systems. Knowledge can play a significant role in establishing links between tasks, robot capabilities, perception, and actions. In this paper, we presented a knowledge-based framework able to atomically transform the robotics tasks into process plans, which are further interpreted into concrete robot commands and executed. The framework incorporates a knowledge base with integrated reasoning services, a planning module able to provide granular robotic plans as well as a sensing module for automatic object recognition. An ontology-based representation of the product is used to connect product designs, robot handling processes, and required equipment as well as to semantically link perception data with geometric features that are represented in a knowledge base. In this context, we also developed an automated mapping between the knowledge base and the planner. The proposed knowledge-driven approach simplifies the programming efforts of the industrial robot, having code generated automatically based on the defined rules, states, and actions. A system engineer then only needs to describe the functionality of the assembly line or characteristics of the product to be assembled, without having to consider further engineering issues. In our application, we successfully used the developed mechanism for planning pick-and-place operations of an

industry robot by Kuka as well as a Festo cartesian robot, which are jointly applied for the assembly of THTs.

In future work, we focus on the introduction of more complex products and production layouts in the framework and integrating temporal planning, as well as re-planning for dynamic environments, and pairing them with a digital twin. We also aim to combine the advantages of knowledge-based systems and deep learning methods. On the one hand, the semantic representation of the object in the environment will be used for the robot to understand and extract information relevant for further actions. On the other hand, the deep learning algorithms will be used to perceive the environment and learn adequate behavior enabling the framework to cope with uncertainties, but also reducing a further need for explicit programming.

List of Figures

List of Listings

List of Tables

Acronyms

CAD Computer-aided design. 14, 21, 27, 36

CPS Cyber-Physical System. 1, 4

CRCL Canonical Robotic Command Language. 20, 32, 38, 39

CURIE Compact URI. 10

GASD Globally Aligned Spatial Distribution. 21–23, 37, 51

HLC High-Level Control. 25, 26

IPC International Planning Competition. 33, 34

JSON JavaScript Object Notation. 10, 39

JSON-LD JSON for Linking Data. 10

LLC Low-Level Control. 25, 26

OWL Web Ontology Language. 3, 8, 11, 12, 26, 29, 31–33, 51

OWL DL Web Ontology Language Description Logic. 12

PCB Printed circuit board. 37, 38, 41, 42

PCL Point Cloud Library. 37

PDDL Planning Domain Definition Language. 3, 16–21, 29, 31–34, 43, 44, 51, 53, 66

PLC Programmable logic controller. 39

RDF Resource Description Framework. 9–12, 32, 53

RDFS RDF Schema. 8, 11, 12

RGB Red Green Blue. 37, 38

RGB-D Red Green Blue - Depth. 36, 37

ROS Robot Operating System. 20, 21

ROSETTA RObot control for Skilled ExecuTion of Tasks. 13, 14, 20, 26

SKOS Simple Knowledge Organisation System. 8

SPARQL SPARQL Protocol and RDF Query Language. 12, 13, 31, 43, 53

SQL Structured Query Language. 13

TCP Tool Center Point. 43, 46

TCP Transmission Control Protocol. 39

THT Through-hole technology. 37, 41–44, 50

UML Unified Modeling Language. 8, 27

URI Uniform Resource Identifier. 10

VOWL Visual Notation for OWL Ontologies. 28, 29

XML Extensible Markup Language. 8, 10, 39

Bibliography

[AGPC16] Angelos Argyrou, Christos Giannoulis, Nikolaos Papakostas, and George Chryssolouris. A Uniform Data Model for Representing Symbiotic Assembly Stations. *Procedia CIRP*, 44:85–90, January 2016.

[AH11] D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Elsevier Science, 2011.

[AKL16] Ron Alterovitz, Sven Koenig, and Maxim Likhachev. Robot planning in the real world: Research challenges and opportunities. *AI Magazine*, 37(2):76–84, Jul. 2016.

[AMQP+13] Ahmed Al-Moadhen, Renxi Qiu, Michael Packianather, Ze Ji, and Rossi Setchi. Integrating robot task planner with common-sense knowledge base to improve the efficiency of planning. *Procedia Computer Science*, 22:211 – 220, 2013. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013.

[ASM17] ASME. *Towards a Robot Task Ontology Standard*, volume Volume 3: Manufacturing Equipment and Systems of *International Manufacturing Science and Engineering Conference*, Jun 2017. ASME 2017 12th International Manufacturing Science and Engineering Conference collocated with the JSME/ASME 2017 6th International Conference on Materials and Processing.

[Bac00] Fahiem Bacchus. Subset of pddl for the aips2000 planning competition. Technical report, The AIPS-00 Planning Competition Comitee, 2000.

[BBB+11] A. Bannat, T. Bautze, M. Beetz, J. Blume, K. Diepold, C. Ertelt, F. Geiger, T. Gmeiner, T. Gyger, A. Knoll, C. Lau, C. Lenz, M. Ostgathe, G. Reinhart, W. Roesel, T. Ruehr, A. Schuboe, K. Shea, I. Stork genannt Wersborg, S. Stork, W. Tekouo, F. Wallhoff, M. Wiesbeck, and M. F. Zaeh. Artificial cognition in production systems. *IEEE Transactions on Automation Science and Engineering*, 8(1):148–174, 2011.

[BCM17] Luca Buoncompagni, Alessio Capitanelli, and Fulvio Mastrogiovanni. A ros multi-ontology references services: Owl reasoners and application prototyping issues. *arXiv preprint arXiv:1706.10151*, 2017.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[CFL+15] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. Rosplan: Planning in the robot operating system. In *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.

[CFP+13] Joel Carbonera, Sandro Fiorini, Edson Prestes, Vitor Jorge, Mara Abel, Raj Madhavan, Angela Locoro, Paulo Gonçalves, Tamas Haidegger, and Craig Schlenoff. Defining position in a core ontology for robotics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1867–1872, 11 2013.

[CM92] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[CPT+16] Matthew Crosby, Ronald P. A. Petrick, César Toscano, Rui Correia Dias, Francesco Rovida, and Volker Krüger. Integrating mission, logistics, and task planning for skills-based robot control in industrial kitting applications. In *PlanSIG*, 2016.

[DADR19] Mohammed Diab, Ali Akbari, Ud Din, and Jan Rosell. Pmk-a knowledge processing framework for autonomous robotics perception and manipulation. *Sensors*, 19, 03 2019.

[DL17] Hui Ding and Chenggang Li. Cyber-Physical System and Its Application in Textile and Chemical Fiber Enterprises. *Open Journal of Social Sciences*, 05:352, October 2017.

[EE16] H. ElMaraghy and W. ElMaraghy. Smart Adaptable Assembly Systems. *Procedia CIRP*, 44:4–13, January 2016.

[EH04] Stefan Edelkamp and Jörg Hoffmann. Pddl2. 2: the language for the classical part of ipc–4. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2004.

[FL03] M. Fox and D. Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, Dec 2003.

[FL06] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27:235–297, 2006.

[GHL+09] Alfonso E. Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners.

Artificial Intelligence, 173(5):619–668, 2009. Advances in Automated Plan Generation.

[GKW+98] Malik Ghallab, Craig Knoblock, David Wilkins, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David Smith, Ying Sun, and Daniel Weld. Pddl - the planning domain definition language. Technical Report Tech Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 08 1998.

[Gru95] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5):907 – 928, 1995.

[HIC13] Jacob O'Donnal Huckaby and Henrik I. Christensen. Toward A Knowledge Transfer Framework for Process Abstraction in Manufacturing Robotics. In *Proceedings of the 30 th International Conference on Machine Learning, Atlanta, Georgia*, volume 28, 2013.

[HJX+18] Xian-Feng Han, Jesse S. Jin, Juan Xie, Ming-Jie Wang, and Wei Jiang. A comprehensive review of 3d point cloud descriptors. *ArXiv*, abs/1802.02297, 2018.

[HLLM19] Timon Hoebert, Wilfried Lepuschitz, Erhard List, and Munir Merdan. Cloud-based digital twin for industrial robotics. In Vladimír Mařík, Petr Kadera, George Rzevski, Alois Zoitl, Gabriele Anderst-Kotsis, A Min Tjoa, and Ismail Khalil, editors, *Industrial Applications of Holonic and Multi-Agent Systems*, pages 105–116, Cham, 2019. Springer International Publishing.

[HLM19] Timon Hoebert, Wilfried Lepuschitz, and Munir Merdan. Automatic ontology-based plan generation for an industrial robotics system. In Peter M. Roth, Andreas Pichler, Robert Sablatnig, Gernot Stübl, and Markus Vincze, editors, *Proceedings of the Joint ARW & OAGM Workshop 2019*, Campus Steyr, Steyr, Austria, 2019 May 9–10 2019. Austrian Robotics Workshop and Austrian Association for Pattern Recognition, University of Applied Sciences Upper Austria.

[HLMM19] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(2):1–187, 2019.

[ILS+19] Ander Iriondo, Elena Lazkano, Loreto Susperregi, Julen Urain, Ane Fernandez, and Jorge Molina. Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning. *Applied Sciences*, 9(2):348, Jan 2019.

[JC16] Binal Javia and Philipp Cimiano. A knowledge-based architecture supporting declarative action representation for manipulation of everyday objects. In *Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering*, MORSE '16, page 40–46, New York, NY, USA, 2016. Association for Computing Machinery.

[JMN16] L. Jacobsson, J. Malec, and K. Nilsson. Modularization of skill ontologies for industrial robots. In *Proceedings of ISR 2016: 47st International Symposium on Robotics*, pages 1–6, 2016.

[JZKS18] Yuqian Jiang, Shiqi Zhang, Piyush Khandelwal, and Peter Stone. Task planning in robotics: an empirical comparison of pddl-based and asp-based systems, 2018.

[KAFZ19] B. Kast, S. Albrecht, W. Feiten, and J. Zhang. Bridging the gap between semantics and control for industry 4.0 and autonomous production. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 780–787, 2019.

[KGJR+16] Evgeny Kharlamov, Bernardo Cuenca Grau, Ernesto Jiménez-Ruiz, Steffen Lamparter, Gulnar Mehdi, Martin Ringsquandl, Yavor Nenov, Stephan Grimm, Mikhail Roshchin, and Ian Horrocks. Capturing industrial information models with ontologies and constraints. In Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web – ISWC 2016*, pages 325–343, Cham, 2016. Springer International Publishing.

[KKSG18] Zeid Kootbally, Thomas R. Kramer, Craig Schlenoff, and Satyandra K. Gupta. Implementation of an ontology-based approach to enable agility in kit building applications. *International Journal of Semantic Computing*, 12(01):5–24, 2018.

[KP20] Ajay Kattepur and Balamuralidhar Purushotaman. roboplanner: a pragmatic task planning framework for autonomous robots. *Cognitive Computation and Systems*, 2:12–22(10), March 2020.

[KSH14] M. Krötzsch, F. Simancik, and I. Horrocks. Description logics. *IEEE Intelligent Systems*, 29(1):12–19, 2014.

[KSL+15] Z. Kootbally, C. Schlenoff, C. Lawler, T. Kramer, and S.K. Gupta. Towards robust assembly with knowledge representation for the planning domain definition language (pddl). *Robotics and Computer-Integrated Manufacturing*, 33:42 – 55, 2015. Special Issue on Knowledge Driven Robotics and Manufacturing.

[LLMN15] Steffen Lohmann, Vincent Link, Eduard Marbach, and Stefan Negru. WebVOWL: Web-based visualization of ontologies. In *Proceedings of EKAW 2014 Satellite Events*, volume 8982 of *LNAI*, pages 154–158. Springer, 2015.

[LZVM11] W. Lepuschitz, A. Zoitl, M. Vallée, and M. Merdan. Toward self-reconfiguration of manufacturing systems using automation agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(1):52–69, 2011.

[McC04] D. McComb. *Semantics in Business Systems: The Savvy Manager's Guide*. The Savvy Manager's Guides. Elsevier Science, 2004.

[McG02] Deborah L McGuinness. Ontologies come of age. *Spinning the semantic web: bringing the World Wide Web to its full potential*, pages 171–194, 2002.

[MHLL19] Munir Merdan, Timon Hoebert, Erhard List, and Wilfried Lepuschitz. Knowledge-based cyber-physical systems for assembly automation. *Production & Manufacturing Research*, 7(1):223–254, 2019.

[OABK+19] Alberto Olivares-Alarcos, Daniel BeÃŸler, Alaa Khamis, Paulo Goncalves, Maki K. Habib, Julita Bermejo-Alonso, Marcos Barreto, Mohammed Diab, Jan Rosell, JoÃ£o Quintas, and et al. A review and comparison of ontology-based approaches to robot autonomy. *The Knowledge Engineering Review*, 34:e29, 2019.

[PBK+16] Frederick Proctor, Stephen Balakirsky, Zeid Kootbally, Thomas Kramer, Craig Schlenoff, and William Shackleford. The canonical robot command language (crcl). *Industrial Robot: An International Journal*, 43:495–502, 08 2016.

[PK15] Mikkel Rath Pedersen and Volker Krüger. Automated planning of industrial logistics on a skill-equipped robot. In *Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, I E E E International Conference on Intelligent Robots and Systems. Proceedings. IROS Hamburg, 2015. null ; Conference date: 02-10-2015 Through 02-10-2015.

[PRK+19] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. Rath Hansen, M. T. Nibe, M. A. Roa, O. Sornmo, S. Gestegard Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kessler, and M. Danzer. Smerobotics: Smart robots for flexible manufacturing. *IEEE Robotics Automation Magazine*, 26(1):78–90, 2019.

[PSRK15] A. Perzylo, N. Somani, M. Rickert, and A. Knoll. An ontology for cad data and geometric constraints as a link between product models and semantic robot task descriptions. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4197–4203, 09 2015.

[RBB09] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ICRA'09, page 1848–1853. IEEE Press, 2009.

[RBTH10] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162, 2010.

[RC11] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[RCH+17] Francesco Rovida, Matthew Crosby, Dirk Holz, Athanasios S. Polydoros, Bjarne Großmann, Ronald P. A. Petrick, and Volker Krüger. Skiros—a skill-based robot control platform on top of ros. In Anis Koubaa, editor, *Robot Operating System (ROS): The Complete Reference (Volume 2)*, pages 121–160. Springer International Publishing, Cham, 2017.

[RFN18] A. Rogalla, A. Fay, and O. Niggemann. Improved domain modeling for realistic automated planning and scheduling in discrete manufacturing. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 464–471, 2018.

[SBF98] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.

[SKKF+19] Veera Ragavan Sampath Kumar, Alaa Khamis, Sandro Fiorini, Joel Luis Carbonera, Alberto Olivares Alarcos, Maki Habib, Paulo Goncalves, Howard Li, and Joanna Isabelle Olszewska. Ontologies for industry 4.0. *The Knowledge Engineering Review*, 34:e17, 2019.

[SM15] Maj Stenmark and Jacek Malec. Knowledge-based instruction of manipulation tasks for industrial robotics. *Robotics and Computer-Integrated Manufacturing*, 33:56 – 67, 2015. Special Issue on Knowledge Driven Robotics and Manufacturing.

[SPM+12] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, and E. Migueláñez. An ieee standard ontology for robotics and automation. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1337–1342, 2012.

[SS10] S. Staab and R. Studer. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer Berlin Heidelberg, 2010.

[ST16] J. P. Silva do Monte Lima and V. Teichrieb. An efficient global point cloud descriptor for object recognition and pose estimation. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 56–63, 2016.

[SZ19] Xiaolei Sun and Yu Zhang. A review of domain knowledge representation for robot task planning. In *Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence*, ICMAI 2019, page 176–183, New York, NY, USA, 2019. Association for Computing Machinery.

[Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer London, 2010.

[TB13] Moritz Tenorth and Michael Beetz. Knowrob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research*, 32(5):566–590, 2013.

[WLRC19] Fei Wang, Chen Liang, Changlei Ru, and Hongtai Cheng. An improved point cloud descriptor for vision based robotic grasping system. *Sensors*, 19:2225, 05 2019.

[WV11] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2987–2992, 2011.

[WVN+19] Bernhard Wally, Jiří Vyskočil, Petr Novak, Christian Huemer, Radek Sindelar, Petr Kadera, Alexandra Mazak, and Manuel Wimmer. Production planning with iec 62264 and pddl. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 492–499, 07 2019.

[WZL07] Shuai Wang, Yu Zhang, and Zhiyong Liao. Review on the knowledge graph in robotics domain. In *3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019)*, pages 424–431. Atlantis Press, 2019/07.

[ZAF16] Stefan Zander, Nadia Ahmed, and Matthias T. Frank. A survey about the usage of semantic technologies for the description of robotic components and capabilities. In *SAMI@iKNOW*, 2016.

Appendix

```
1  (define (domain KnowDrift)
2
3   (:requirements :strips :typing)
4
5   (:types
6          cad:Pose ro:WorkingRange ro:PhysicalObject - object
7          ro:Device ro:Part - ro:PhysicalObject
8          ro:Gripper ro:Robot - ro:Device
9   )
10
11  (:predicates
12   (ro:isSubDeviceOf ?childDevice - ro:Device ?parentDevice - ro:Device)
13   (:canHandle ?gripper - ro:Gripper ?object - ro:PhysicalObject)
14   (:hasPose ?physicalObject - ro:PhysicalObject ?pose - cad:Pose)
15   (:isCurrentlyWithin ?robot - ro:Robot ?range - ro:WorkingRange)
16   (:isHandling ?gripper - ro:Gripper ?object - ro:PhysicalObject)
17   (isHandlingPart ?gripper - ro:Gripper)
18   (:canReach ?robot - ro:Robot ?pose - cad:Pose)
19   (:containsPose ?range - ro:WorkingRange ?pose - cad:Pose)
20   (:canBePlacedAt ?object - ro:PhysicalObject ?pose - cad:Pose)
21  )
22
23  (:action pick
24    :parameters (?robot - ro:Robot ?gripper - ro:Gripper
25     ?part - ro:PhysicalObject ?pose - cad:Pose ?range - ro:WorkingRange)
26
27    :precondition (and (:hasPose ?part ?pose)
28     (:canReach ?robot ?pose)
29     (:isCurrentlyWithin ?robot ?range)
30     (ro:isSubDeviceOf ?gripper ?robot)
31     (:canHandle ?gripper ?part)
32     (not (isHandlingPart ?gripper))
33     (:containsPose ?range ?pose)
34    )
35
36    :effect (and (:isHandling ?gripper ?part)
37     (isHandlingPart ?gripper)
38     (not (:hasPose ?part ?pose))
39    )
40  )
41
```

```
42   (:action move
43     :parameters (?robot - ro:Robot ?from - ro:WorkingRange
44      ?to - ro:WorkingRange)
45
46     :precondition (:isCurrentlyWithin ?robot ?from)
47
48     :effect (and (:isCurrentlyWithin ?robot ?to)
49      (not (:isCurrentlyWithin ?robot ?from))
50     )
51   )
52
53   (:action drop
54     :parameters (?robot - ro:Robot ?gripper - ro:Gripper
55      ?part - ro:PhysicalObject ?pose - cad:Pose ?range - ro:WorkingRange)
56
57     :precondition (and (:canReach ?robot ?pose)
58      (ro:isSubDeviceOf ?gripper ?robot)
59      (:isHandling ?gripper ?part)
60      (:isCurrentlyWithin ?robot ?range)
61      (:containsPose ?range ?pose)
62      (:canBePlacedAt ?part ?pose)
63     )
64
65     :effect (and (:hasPose ?part ?pose)
66      (not (:isHandling ?gripper ?part))
67      (not (isHandlingPart ?gripper))
68     )
69   )
70
71   (:action pass
72     :parameters (?senderRobot - ro:Robot ?receiverRobot - ro:Robot
73      ?senderGripper - ro:Gripper ?receiverGripper - ro:Gripper
74      ?part - ro:PhysicalObject ?pose - cad:Pose ?range - ro:WorkingRange)
75
76     :precondition (and (:canReach ?senderRobot ?pose)
77      (:canReach ?receiverRobot ?pose)
78      (:isCurrentlyWithin ?senderRobot ?range)
79      (:isCurrentlyWithin ?receiverRobot ?range)
80      (:containsPose ?range ?pose)
81      (ro:isSubDeviceOf ?senderGripper ?senderRobot)
82      (ro:isSubDeviceOf ?receiverGripper ?receiverRobot)
83      (:canHandle ?senderGripper ?part)
84      (:canHandle ?receiverGripper ?part)
85      (:isHandling ?senderGripper ?part)
86      (not (isHandlingPart ?receiverGripper))
87     )
88
89     :effect (and (:isHandling ?receiverGripper ?part)
90      (isHandlingPart ?receiverGripper)
91      (not (:isHandling ?senderGripper ?part))
92      (not (isHandlingPart ?senderGripper))
93     )
94   )
95 )
```

Listing 1: The full PDDL-domain of the pick-and-place application use-case.