2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). Mexico City, Mexico. October 25-27, 2023

ROS-driven Disassembly Planning Framework incorporating Screw Detection

Timon Hoebert
Practical Robotics Institute Austria
Vienna, Austria
hoebert@pria.at

David Neubauer
TU Wien
Vienna, Austria
e11718344@student.tuwien.ac.at

Munir Merdan
Practical Robotics Institute Austria
Vienna, Austria
merdan@pria.at

Wilfried Lepuschitz
Practical Robotics Institute Austria
Vienna, Austria
lepuschitz@pria.at

Stefan Thalhammer
TU Wien
Vienna, Austria
thalhammer@acin.tuwien.ac.at

Markus Vincze
TU Wien
Vienna, Austria
vincze@acin.tuwien.ac.at

***Abstract*—The automation of electronic waste disassembly processes is very challenging due to the diversity and conditions of the products, but also due to the heterogeneous disassembly environments based on different hardware and software components. All these resources and the subjacent information flows should be coordinated and integrated to ensure an effective disassembly process. In this paper, we present a developed architecture for closed-loop planning and controlling dynamic disassembly processes of robots. We extended the infrastructure provided by the Robot Operating System (ROS) to integrate the components of the robotic system with its vision system and the software components for inference, replanning, and knowledge transfer. The architecture was implemented for a real use case of antenna amplifier disassembly. The implemented framework is generalizable for other purposes implementing four automatic configuration mechanisms to support domain-specific tailoring: code generation, object serialization, object mapping, and object-triple mapping.**

***Index Terms*—disassembly, robot, vision system, ROS, ontology, planning, PDDL, OWL, screw detection, artificial neural networks**

I. Introduction

Disassembly is a very complex and resource extensive process, which focuses on extracting valuable components from waste electrical or electronic equipment (WEEE). Despite the potential for automation to enhance efficiency, it has yet to reach its full potential due to various issues that restrain the robotization of this process [19]. On the one hand, the process is challenging with the variability and uncertainties concerning the type and conditions of the product [11]. On the other hand, the process integrates various operational challenges such as the recognition of components, collision-free manipulation, and tooling as well as collaboration issues related to human-machine interaction. Moreover, in such a heterogeneous environment, the involved resources operate with various types of data that can also restrain the effective accomplishment of the disassembly process. In this context, the seamless information flow and synchronization of all involved subsystems is required to ensure robust robotic systems function in such a collaborative and dynamic environment.

Several frameworks have been proposed for supporting the development and operations of robotic systems such as Microsoft's Robotics Developer Studio, Open Robot Control Software (OROCOS), Orca, or the Robot Operating System (ROS) [1], [10]. ROS [14] is one of the most popular frameworks for the development of robotics systems. It facilitates the sharing of data between different components through a network of nodes, which communicate by exchanging defined messages using a publish/subscribe mechanism. Such kind of architecture is easily extensible since new nodes can be introduced without changing anything in the rest of the implementation.

Besides, various tools and services are available for the framework to support trajectory planning, robot control, perception, simulation, etc. For instance, a ROS-based architecture is used for supporting the collaboration between humans and robots in a shared workspace scenario of electronic vehicle battery quality checking [27] as well as in a virtual hybrid assembly cell [25]. Moreover, ROS is also used for the sensor-guided assembly of full-scale composite blade panels with human-assisted path planning [18] as well as to implement a visual inspection system in a robot work cell [13]. Nevertheless, we haven't found any implementation focused on disassembly processes utilizing the advantages of the ROS-provided infrastructure for interoperability coordination between the involved sub-systems. Various types of products have been employed as use cases for robot-driven disassembly. For instance, approaches for electric vehicle battery disassembly were presented by Choux et al. [2] and Meng et al. [17]. Robotic systems for the automated dismantling of PCB waste were introduced by Doroftei et al. [2] and for LCD screens by Vongbunyong et al. [26] as well as Foo et al. [6].

In our work, we present a knowledge-driven robotic framework focused on automatizing disassembly processes with

The authors acknowledge the financial support from the "ICT of the Future DE-AT AI" program of the Austrian Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology as well as German Federal Ministry for Economic Affairs and Energy under contract FFG 887636

979-8-3503-0676-7/19/$31.00 ©2023 European Union

antenna amplifiers as a use case. These types of amplifiers are located on cell phone towers and are commonly replaced every few years with the introduction of new mobile phone standards. The approach is based on our previous research, where we focused on planning pick-and-place operations during the assembly of PCBs [12]. The currently developed framework integrates an ontology with the high-level decision-making component, which should provide autonomy and flexibility for robot actions. It also integrates a vision system that is used for component identification as well as for searching specific components (screws) during the planning process.

The rest of the paper is organized as follows: In Section II, we present the architecture of the framework. Section III describes the vision system and an evaluation of the approach. Section IV is focused on the implementation and data flow. Finally, Section V concludes the paper and presents an outlook on future research.

II. Architecture

The architecture of the system consists of four main components, as visualized in Figure 1: the ontology, the Decision Making, the vision system, and the robot hardware. The ontology stores all explicit knowledge and derives new implicit knowledge. The Decision Making Component is capable of generating efficient plans based on the current ontology information. The vision system detects specific objects and reports them back to the ontology via the Decision Making component. During plan execution, the robot hardware executes planned actions and notifies the Decision Making about execution feedback, so that in case of an execution failure a replanning can be triggered.

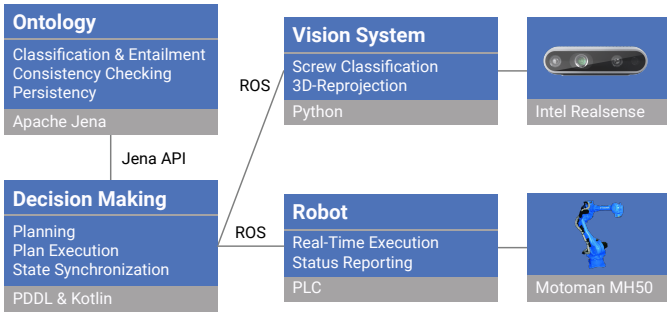

Fig. 1: **The system architecture** with its four main components, their functions, and their interfaces.

A. Ontology

The implemented ontology contains relevant information about the current state and the goal state of the system. This includes data about physical objects, such as the robot and its tools, as well as application-specific information, such as screws and disassembly status information. The ontology fulfills three tasks: persistency, classification and entailment as well as consistency checking. The ontology persists all information in triple statements which are assigned to a subdividing graph. For this purpose, the information is stored using the OWL[1] standard which is based on RDF[2]. For planning, there are two important graphs, one for storing the current state and one for the goal state. This knowledge is either statically implemented in the ontology or dynamic information from external system components, such as the vision system and the robot hardware (see Section IV).

RDF enables the use of semantic reasoning for classification and consistency checking. Especially the manually entered static knowledge can be checked for consistency using a reasoner. For this purpose, our system relies on the Pellet Reasoner [21]. It is also incorporated to automatically deduce new knowledge within our ontology such as transitive relations or to abstract the detailed knowledge to high-level information for computational efficiency during planning.

B. Decision Making

The Decision Making is the central component for using implicit and explicit ontology information to generate an efficient plan to transition from the current state to the goal state. In our case, the goal state is a disassembled object with detached connections (e.g. loosened screws). For this purpose, we employ the Planning Domain Definition Language PDDL[3] for modeling planning domains and problems, where also multiple solvers exist.

In previous work, we implemented a planning engine on top of PDDL to automate the plan generation, parsing, and execution for single execution to execute robotic assembly operations [12]. We also implemented a scheme to directly map ontology triple statements with PDDL state information, the so-called PDDL-predicates. The novelty of this work is the adaption of this framework to cope with dynamic environments, such as in the disassembly domain, where replanning is necessary for unexpected situations. The planning and replanning procedure of the adapted framework is visualized in Figure 2.

The input of the system is the PDDL planning domain which is parsed in the first step. Most notably, it contains a list of possible parametrizable actions and their state modifications. For example, the disassembly use case comprises the unscrewing action, which modifies the connection status of screws. For planning, besides the general PDDL domain which contains the planning instance information, the PDDL problem is needed. The default assumption of the Decision Making classifies all predicates of the PDDL domain as relevant and all triple statements using these predicates should be retrieved. Therefore, it automatically uses the PDDL predicates to generate queries for these relevant statements. For example, because of the listed unscrewing action within the PDDL-Domain, only screw and screw status information is considered to be relevant for the planning process, and only this information is queried from the ontology and sent to the planner. These RDF statements are subsequently mapped to PDDL. During this process, all occurring instances are recorded to query the

[1]Web Ontology Language: https://www.w3.org/OWL/
[2]Resource Description Framework: https://www.w3.org/RDF/
[3]PDDL: https://planning.wiki/ref/pddl/domain

types of instances which are needed for the PDDL Planner, for example, if a resource is of type screw.

Since the target language PDDL only supports single-inheritance but the OWL-source multi-inheritance, only the most specific types are computed using a matching procedure of the deepest type within its hierarchy tree. Within the disassembly domain, an object of type screw is also typed as a connector, which is also typed as a physical object. For planning, only the most concrete type, which is deepest within the class hierarchy, is computed.

With the PDDL domain and the mapped PDDL problem, the planner computes multiple solutions for the problem using a specified optimization function. In many cases, just as in the disassembly use case, the optimization function minimizes the makespan, which is the execution time of all robot operations. Using the optimal computed plan, its actions and parameter resources are parsed to start the plan execution. As also shown in Figure 7, the Plan Executor notifies the application logic about the execution of each individual plan action with the planned parameter resources. Before each action execution, the application logic can query additional properties of resource parameters. For example, the planner only computes an unscrewing action for a named object without any properties. For plan computation efficiency, the exact head type is neglected, but for execution on the robot hardware, this information must be queried beforehand.

If the plan action execution of the application logic succeeds, the new ontology state is automatically updated by the plan executor using the integrated data from the PDDL domain and the plan resource parameters. The application logic needs to be implemented only for execution failure and unexpected state updates.

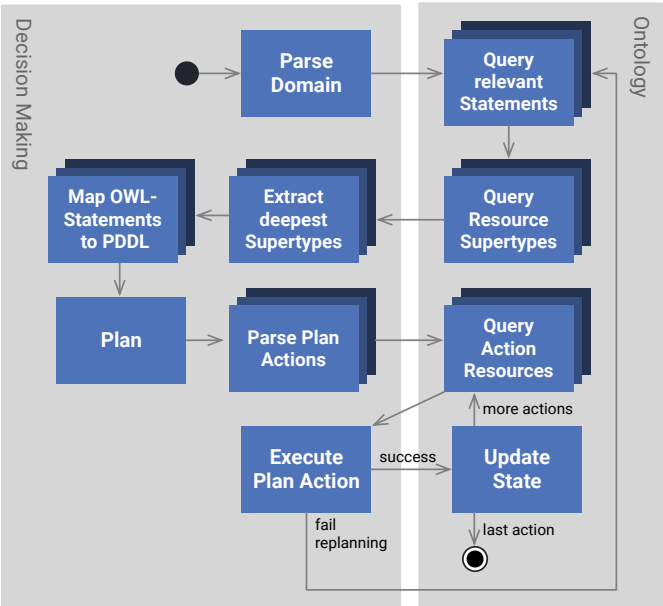

Fig. 2: **The (re-)planning and execution procedure** of the Decision Making component (left column) with its interactions with the ontology (right column).

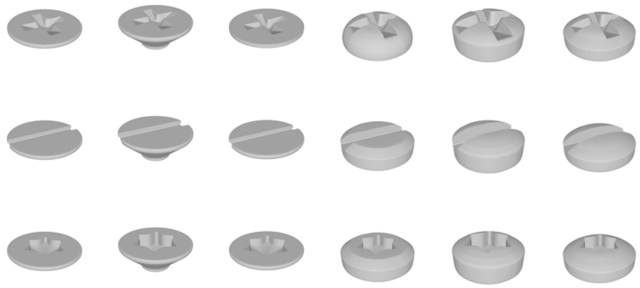

Fig. 3: **Screw models for rendering** 3D screw models used for training dataset creation. The three screw categories (Phillips, Slotted, Torx) are shown from top to bottom, whereas the 6 different variants (cone and dome, each with a normal, lifted, and a wide version) are shown.

III. Vision System

The purpose of the vision system is the detection and $6D$ pose estimation of the screws that fasten the top layer of the antenna amplifier. RGB-D images are used as input. RGB images are used for extracting sparse location priors for the individual screw instances. Depth images are used for $3D$ translation estimation using the $3D$-reprojection of the detected screw locations, and for $3D$ rotation estimation by fitting a plane on the antenna receiver using the depth image. The primary challenge is to robustly handle the expected test time variations. On the one hand, screw instances are unknown during training time and potentially vary largely during test time. On the other hand, the expected places of deployment are very dynamic with respect to illumination and camera viewpoints, and the antenna amplifiers have to be assumed to be in various states of ageing and contamination with dirt. In order to robustly handle these challenging conditions we use domain randomization for diversifying object models and image variations [24].

A. Data Creation

To robustly handle the expected test time variations we rely on a heavily diversified rendered training dataset for training object detectors which are Convolutional Neuronal Networks. To handle unknown screw instances we create screw priors. The expected screw categories are: Phillips, Slotted, and Torx. For each, 6 different 3D models are created and randomized with respect to head type, height and width, as shown in Figure 3.

To handle different scene configurations, antenna amplifier aging and contamination, rendering parameters such as screw poses and texture, illumination, and background are randomized for training data creation with BlenderProc [4]. In order to robustify the trained detectors against diverse antenna amplifier appearances, we utterly decouple training data creation from the relevant object, i.e. the antenna amplifier, by using the Cut-Paste strategy [5]. We randomly sample between 10 and 50 of the created 3D models per rendered image. Object

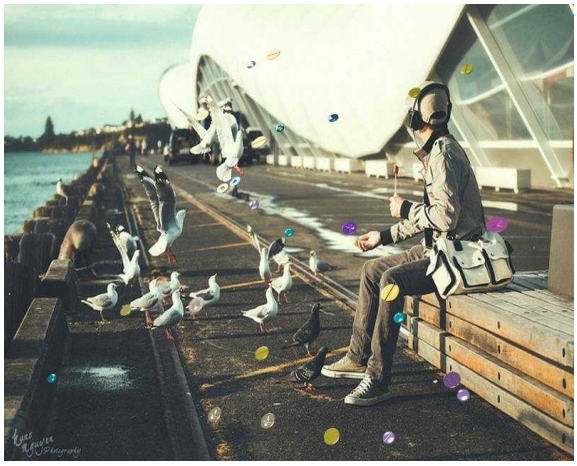

Fig. 4: **Exemplary training image** Randomized rendered training image. The random aspects are screw location and texture, camera location, light location and strength and background.

models are placed on a plane, with the screws' locations, head diameters, and textures randomized. Texture randomization is applied to a) account for various degradation states of the screw, e.g. rust, and b) to bias object detector training toward geometry [7]. The scene is illuminated by two randomly placed point lights facing the center of the scene. The strength of each light is varied independently for each image. Images are captured by a randomly place camera facing the scene's center. Figure 4 shows one of the 500 unique renderings.

For validation purpose we split the rendered dataset in 450 images for training and 50 for validating the training progress. We additionally validate the best performing detector, FCOS, trained on rendered data, on a real test dataset. This dataset consist of 159 images, which are annotated with screw category, $2D$ bounding box, per-instance mask, and $6D$ pose. For annotation, 3D-DAT is used [22].

B. Screw Detection

The aforementioned rendered training set is used for training and comparing different object detectors. Table I presents a comparison of RCNN [20], Faster RCNN [8], and FCOS [23], all fine-tuned with the created renderings, using mean average precision (mAP) as metric. All detectors are used with the loss functions presented in the respective publications, with ResNet50 [9] backbone pre-trained on ImageNet [15], and are fine-tuned for 12 epochs on the rendered test dataset using the Adam optimizer with a learning rate of 0.01. During training, images are augmented by randomly shifting channel intensities in RGB and HSV space, and by randomly changing brightness and contrast. The upper part of the table shows that FCOS achieves the best results on the rendered validation set. Comparing the performance of FCOS on the rendered and the real data shows that no drop in bounding box estimation accuracy occurs when validating on real data. This indicates that the presented domain randomization scheme generalizes

Model	Backbone	Validation	mAP
RCNN [20]	ResNet50	Rendered	0.5
Faster-RCNN [8]	ResNet50	Rendered	0.6
FCOS [23]	ResNet50	Rendered	0.8
FCOS [23]	ResNet50	Real	0.8

TABLE I: **Screw detection evaluation** Detection performance in comparison, using mean average precision (mAP) for quantification.

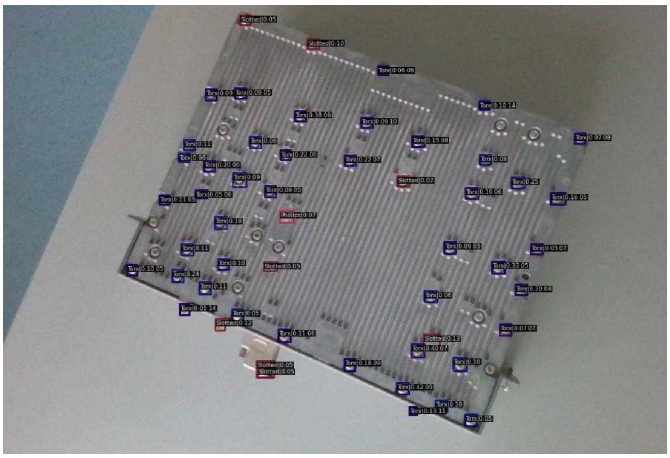

Fig. 5: **Screw detection example** Detections on a real validation image using FCOS. The blue bounding boxes show correctly detected and classified screws. Incorrect classifications are shown in red.

effectively to the real world test case. Figure 5 shows an exemplary image of screw detections.

C. 6D Screw Pose Estimation

For $6D$ screw pose estimation we exploit the sparse location priors generated by the object detector and the observed depth image. The center point of the detected bounding box and the observed depth image are used for retrieving the screw's depth. Using this depth and again the center point the $3D$ screw location is retrieved using $3D$-reprojection equation.

In order to retrieve the $3D$ screw rotations we use RANSAC-based plane fitting for normal estimation. The estimated antenna amplifier surface normal is used as $3D$ rotation for each of the fastened screws. We use RANSAC with three samples, an inlier threshold 0.01 meters, a expected probability of 0.999, and 1000 iterations. Pose estimation results are shown with the point cloud in Figure 6.

IV. Implementation

The proposed system architecture is modularly conceptualized with a distinct separation between the domain-specific application logic and the generalized framework. This way, the system is adaptable for implementing new application logic for different planning scenarios. For this purpose, the proposed framework uses four automatic configuration mechanisms to support this domain-specific implementation: code generation,

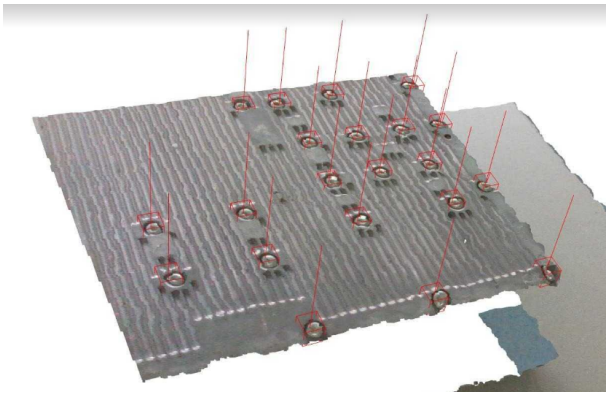

Fig. 6: **Visualization of 6D poses** $3D$ translations are derived from the screw detections and the observed depth image. $3D$ rotations are derived using plane fitting.

object serialization, object mapping, and object-triple mapping. Figure 7 gives an overview of these mechanisms and their interactions within the system.

Using these configuration mechanisms, the application logic only needs to implement two domain-specific data wrappers, the transfer objects to communicate with other hardware components, and persistency objects to store knowledge in the ontology for persistency, reasoning, and automated planning.

The application logic interacts with the hardware components via ROS [14] using either a synchronous request/response modality (ROS services) or an asynchronous publish/subscribe mechanism (using ROS messages). ROS uses a standardized interface definition language to specify ROS messages and ROS services. Listing 1 shows the ROS message for a screw with its pose (the position and rotation), and other attributes, such as the head type and size. In our case, the vision system publishes detected screws whereas the Graphical User Interface (GUI) subscribes to the Application Logic to inform and visualize current operations. On the other hand, the robot hardware needs synchronization with the Decision Making when a hardware operation is finished and its response is awaited. Therefore each robot operation is implemented using a ROS service. It defines a set of detected screws to unscrew and replies with an individual status response code for each screw. In case, a single screw is broken or too tight for automated handling, these responses are replied to trigger replanning.

These ROS message and service definitions are used as input by the framework to automatically generate the transfer classes for the application logic. Their instantiated transfer object are automatically serialized and deserialized to ROS messages which are transmitted to the ROS network using ROS-Bridge [3]. We published this part of the framework into a separate library for general usage[4].

For persistency and planning, the application logic needs to implement persistency objects which are automatically stored

[4]Kotlin ROS Bridge code generator and serializer: https://github.com/thoebert/krosbridge

```
geometry_msgs/Pose pose
string head_type #Philips, Torx, Slotted
bool magnetic
float32 head_size
float32 length
float32 detection_accuracy
```

Listing 1: **Screw ROS-Message definition** with a listing of all attributes and their datatypes.

in the ontology using object-triple mapping [16]. In many applications, the transfer objects need to be stored and used for planning. For example, this applies to the screw transfer classes, since they are received from the vision system and need to be persisted in the ontology for further planning. In this case, the persistency objects can be similarly implemented and directly converted to the persistency objects using object-object mapping.

V. Conclusion

In this paper, we described a developed architecture for electronic waste disassembly with the use case of antenna amplifier disassembly. The architecture integrates an ontology-based decision-making component with a vision system and its robot hardware. Linking the ontology with a vision system provides interoperability in the dynamic and heterogeneous disassembly environment and enables that the information of a captured image can be understood and used for manipulation planning of each disassembly task. The vision system, trained using a generated synthetic dataset, can detect screw heads and achieves an 80% mean average precision on a real-world validation dataset with antenna amplifier objects as use-case. The whole system operates in a closed feedback loop to cope with unforeseen disassembly exceptions by incorporating hardware feedback into the ontology for replanning. The architecture was implemented in ROS and facilitates four mechanisms to be easily extended for additional hardware components as well as other use cases. In future work, we aim to introduce humans into the environment and enable a collaborative human-robot disassembly process.

References

[1] Aakash Ahmad and Muhammad Ali Babar. Software architectures for robotic systems: A systematic mapping study. *Journal of Systems and Software*, 122:16–39, 2016.

[2] Martin Choux, Eduard Marti Bigorra, and Ilya Tyapin. Task planner for robotic disassembly of electric vehicle battery pack. *Metals*, 11(3):387, Feb 2021.

[3] Christopher Crick, Graylin Jay, Sarah Osentoski, Benjamin Pitzer, and Odest Chadwicke Jenkins. *Rosbridge: ROS for Non-ROS Users*, pages 493–504. Springer International Publishing, Cham, 2017.

[4] Maximilian Denninger, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82):4901, 2023.

[5] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1301–1310, 2017.

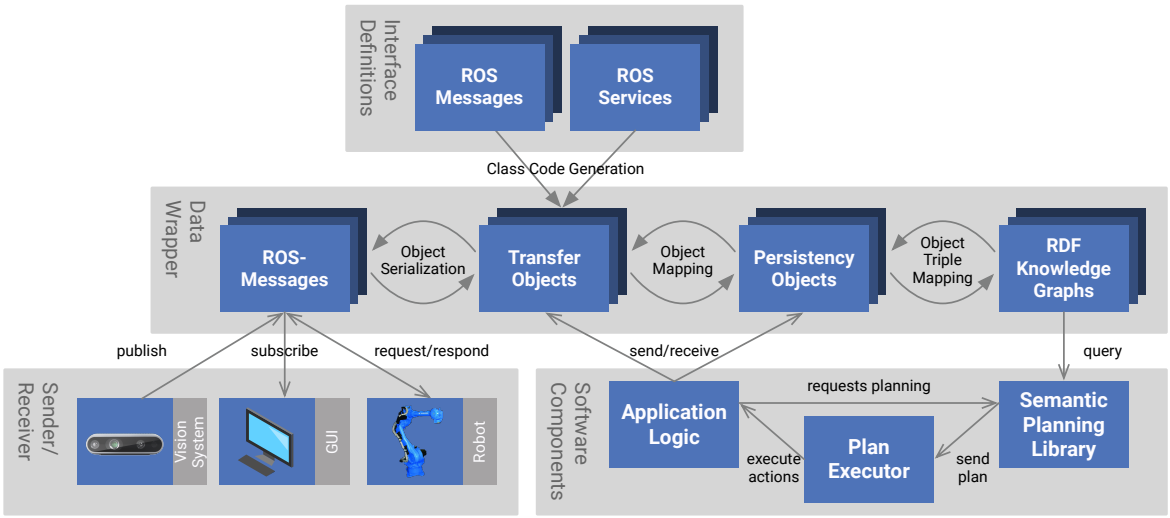

Fig. 7: **Automated configuration mechanisms and data transfer within the system.** The central application logic can either transfer objects to communicate with ROS components or use wrapping persistency objects to store data in the knowledge graph. In the case of replanning, only the persisted knowledge graph information is used by the planner. On the other side, the transfer objects are auto-generated using the ROS interface definitions and can be automatically mapped to persistency objects for storage.

[6] Gwendolyn Foo, Sami Kara, and Maurice Pagnucco. An ontology-based method for semi-automatic disassembly of lcd monitors and unexpected product types. *International Journal of Automation Technology*, 15(2):168–181, 03 2021.

[7] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[8] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Abdelfetah Hentout, Abderraouf Maoudj, and Brahim Bouzouia. A survey of development frameworks for robotics. In *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*, pages 67–72, 2016.

[11] Sebastian Hjorth and Dimitrios Chrysostomou. Human–robot collaboration in industrial environments: A literature review on non-destructive disassembly. *Robotics and Computer-Integrated Manufacturing*, 73:102208, 2022.

[12] Timon Hoebert, Wilfried Lepuschitz, Markus Vincze, and Munir Merdan. Knowledge-driven framework for industrial robotic systems. *Journal of Intelligent Manufacturing*, 34(2):771–788, 2023.

[13] Tatyana Ivanovska., Simon Reich., Robert Bevec., Ziga Gosar., Minija Tamousinaite., Ales Ude., and Florentin Wörgötter. Visual inspection and error detection in a reconfigurable robot workcell: An automotive light assembly example. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VCEA*, pages 607–615. INSTICC, SciTePress, 2018.

[14] Anis Koubâa et al. *Robot Operating System (ROS).*, volume 1. Springer, 2017.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[16] Martin Ledvinka and Petr Křemen. A comparison of object-triple mapping libraries. *Semantic Web*, 11:483–524, 2020. 3.

[17] Kai Meng, Guiyin Xu, Xianghui Peng, Kamal Youcef-Toumi, and Ju Li. Intelligent disassembly of electric-vehicle batteries: a forward-looking overview. *Resources, Conservation and Recycling*, 182:106207, 2022.

[18] Yuan-Chih Peng, Shuyang Chen, Devavrat Jivani, John Wason, William Lawler, Glenn Saunders, Richard J. Radke, Jeff Trinkle, Shridhar Nath, and John T. Wen. Sensor-guided assembly of segmented structures with industrial robots. *Applied Sciences*, 11(6), 2021.

[19] Hendrik Poschmann, Holger Brüggemann, and Daniel Goldmann. Disassembly 4.0: A review on using robotics in disassembly tasks as a way of automation. *Chemie Ingenieur Technik*, 92(4):341–359, 2020.

[20] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.

[21] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007. Software Engineering and the Semantic Web.

[22] Markus Suchi, Bernhard Neuberger, Amanzhol Salykov, Jean-Baptiste Weibel, Timothy Patten, and Markus Vincze. 3d-dat: 3d-dataset annotation toolkit for robotic vision. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9162–9168. IEEE, 2023.

[23] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9626–9635, 2019.

[24] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[25] Panagiota Tsarouchi, Sotiris Makris, George Michalos, Alexandros-Stereos Matthaiakis, Xenofon Chatzigeorgiou, Athanasios Athanasatos, Michael Stefos, Panagiotis Aivaliotis, and George Chryssolouris. Ros based coordination of human robot cooperative assembly tasks-an industrial case study. *Procedia CIRP*, 37:254–259, 2015. CIRPe 2015 - Understanding the life cycle implications of manufacturing.

[26] Supachai Vongbunyong, Sami Kara, and Maurice Pagnucco. Learning and revision in cognitive robotics disassembly automation. *Robotics and Computer-Integrated Manufacturing*, 34:79–94, 2015.

[27] Pai Zheng, Shufei Li, Liqiao Xia, Lihui Wang, and Aydin Nassehi. A visual reasoning-based approach for mutual-cognitive human-robot collaboration. *CIRP Annals*, 71(1):377–380, 2022.