Virtual Teaching for Assembly Tasks Planning

Alexandre Angleraud[1], Robert Codd-Downey[2], Metodi Netzev [1], Quentin Houbre [1] and Roel Pieters[1]

***Abstract*— Small and Medium-size Enterprises require increasingly versatile robots that are capable of learning new skills during their operating life span, in addition to the ability to integrate themselves into human teams as a new and effective partner. Teaching information and skills to a robot can quickly become very complex, especially when considering that the human partner of the robot is inexperienced in the field and does not have access to intuitive interaction channels to train such robots. In this work, a system is proposed that can virtually define an assembly task, based on CAD modelling that describe constraints between assembly parts. These constraints are then extracted in an ontology which is automatically translated into Simple Temporal Networks (STNs) leading to joint action plans.**

Results show the automatic translation between virtual teaching and knowledge acquired by the robot on a peg-in-hole problem before illustrating the connection to planning for human-robot collaborative tasks. As a proof of concept, these developments demonstrate that a CAD guided assembly planner circumvents the need for skilled robot programming.

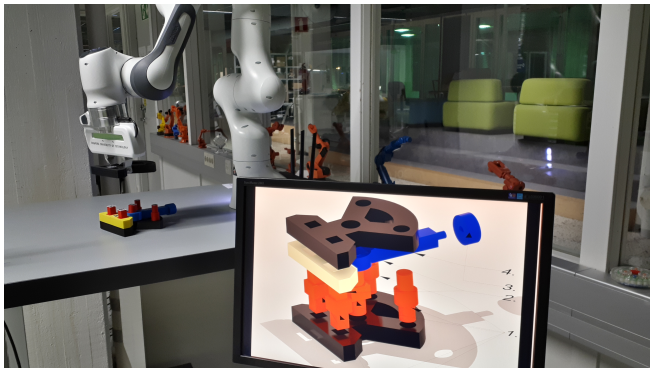

Fig. 1. CAD-guided assembly planning. The virtual teaching process starts from a CAD model that includes all assembly parts and virtually specifies all steps in the assembly task (displayed on a screen with the Cranfield benchmark). Following, this is stored in an ontology to generate task plans for a collaborative robot arm (Franka Emika's Panda).

I. INTRODUCTION

Small and Medium-size enterprises (SMEs) are shifting from mass production to smaller batch production to facilitate developmental changes in manufacturing [1]. To this end, the traditional industrial robot, fast and robust but also heavy and stiff, is being replaced by smaller robots with a design aimed towards human collaboration in a safe and effective manner. These collaborative robots (cobots) are teammates to which new skills will also need to be taught. Teaching operators new skills or tasks is a routine activity, yet not as straightforward when considering human-robot interaction with cobots. Many underlying details, such as action sequences or part orientation, are crucial but not necessarily intuitive to describe to a machine. Since it cannot be expected that operators in factory floors have a deep understanding of the robot's internal functioning, nor the capability to program them, methods need to be sought to enable fast and intuitive ways for communication.

Nowadays, robots have integrated software components that allow the quick automation of repetitive tasks. Unfortunately, this ignores the context and semantics of what is being executed, thereby limiting the usability within a human-robot team. In other words, it would be very hard for such a robot to be included in a team for constructive collaboration, while ensuring efficiency and safety.

Ontologies offer a solution to store semantic information, enable systematic reasoning and can be edited to add knowledge to the robot during its operating life span. Nevertheless, ontologies remain complex to handle and require tools for intuitive content management. In this paper we explore virtual teaching for assembly tasks (see Figure 1).

The contributions in this work are as follows:

- A CAD-guided modelling approach to specify an assembly task and its constraints,
- Automatic generation of semantic task knowledge in an ontology, such as assembly parts, assembly part properties and links between parts,
- Utilizing this semantic task knowledge for planning and executing the assembly task in human-robot collaboration.

Section II will review the main concepts involved in our system. Section III describes the teaching phase before connecting it to planning in Section IV. Section V then specifies the considered assembly task, the Cranfield Assembly Benchmark [2]. Finally, Section VI discusses the approach and its limitations, and the remaining points for future work.

II. RELATED WORK

This work aims at facilitating the transfer of knowledge to a robot to enable human-robot teams to perform joint actions together. Teaching is done through virtual demonstrations, operated on a CAD model of the assembly task. First, a review of dynamic planning, semantic systems and collaboration mechanisms in human-robot teams is given to familiarize the reader with the concepts involved in the process.

A. Human-Robot collaboration

There exists several levels of collaboration (coexistence, assistance, cooperation, etc.) [3] but regardless, teammates

[1]Unit of Automation Technology and Mechanical Engineering, Tampere University, Tampere, Finland; firstname.surname@tuni.fi

[2]Department of Electrical Engineering and Computer Science, York University, Toronto, Canada; robert@eecs.yorku.ca

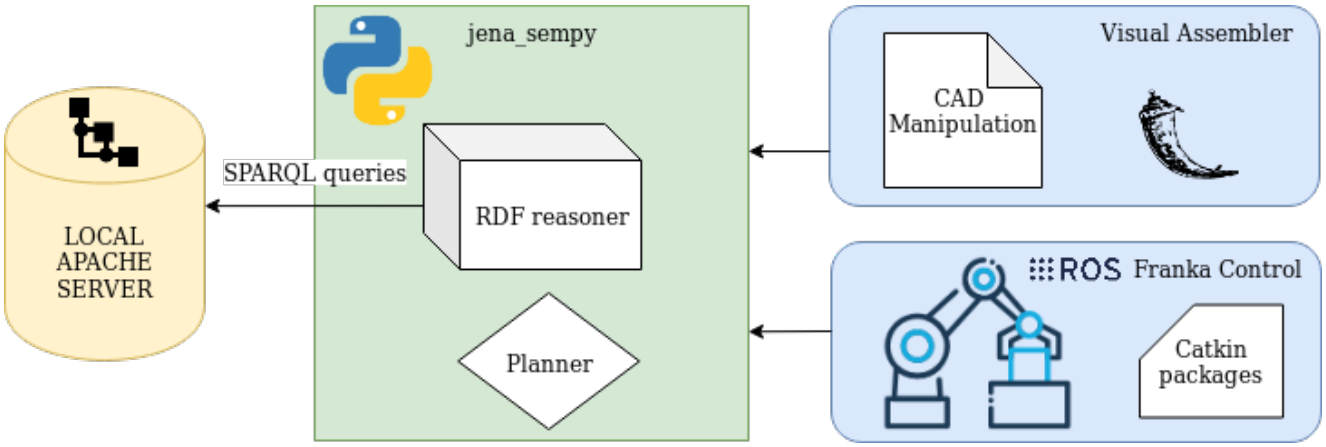

Fig. 2. Software architecture. Composed of 4 parts. A generic library for semantic reasoning and planing. An ontology server based on Apache Jena Fuseki. A Flask web server allowing to teach the robot new tasks. ROS packages for translating this plan into actions that the robot can execute. It is distributed on github and included into Docker containers.

need to take each other's action into account [4]. Moreover, to successfully collaborate both need to share the same understanding of the task at hand to know what the other is about to do. In that sense communication is an important factor for a successful task. It has been shown that bottom-up solutions in which the robot acquires by itself a model of knowledge and develops the capacity to communicate about it [5] can lead to encouraging results. However, we make use of semantics [6] to bring predictability to our system.

B. Semantic systems

The previously mentioned planning strategies rely upon knowledge. Knowledge that can be stored in models [7] or ontologies for example [8]. An ontology is an XML file gathering semantic knowledge about the robot, its environment and the tasks it's supposed to accomplish. This knowledge can be further altered to teach the robot new skills [9]. Other methods include using the sensorimotor information to physically demonstrate tasks [10], [11], and as we will explore in this paper, 3D CAD models. They can be used to learn low level information such as controllers and motion plans in autonomous assembly tasks [12] or in collaboration [13]. They can also be combined with fuzzy logic to infer some parameters of the assembly [14]. In our case we use the CAD model to deduce high level knowledge, and add it to the robot knowledge base. Another important high-level consideration in collaborative tasks is the task allocation problem that can also make use of its own models [15]. We address simultaneously task allocation and scheduling within the Simple Temporal Network formalism on which we come back later.

C. Dynamic planning

Once specified, a task can either be performed in any order and will not require any action selection process. Or, if there exist dependencies between sub-sequences, it will have to be performed in a precise sequence that the robot needs to be aware of. This type of tasks requires additional dynamic planning features in a collaborative context. Contrarily to traditional planning where everything has to be specified beforehand, dynamic planning allows for more flexibility as it takes into account the evolution of the environment and allows the robot to react to them. Several ways to conceive planning exists, see e.g. [16] or [17], where a Simple Temporal Network (STN) is built to answer planning and scheduling. It is based on the Simple Temporal Problem defined in [18]. STNs have since then been extended in various ways [19]. An important challenge is to build those plans automatically [20]. The system presented in this paper uses knowledge stored in ontologies to do so.

III. VIRTUAL DEMONSTRATION

To endow the robot with the ability to perform new tasks, a description of this task needs to be made available to the knowledge base. Writing such a description by hand can quickly become cumbersome, even when using tools such as the Protégé editor[1]. Writing an ontology requires a high degree of expertise and a step learning curve for the end user. We propose an approach to allow the user to virtually specify the assembly task and it's properties and have them automatically translated into an ontology (see Figure 2). This section describes the process of teaching the robot a new task using the framework.

A. Modelling and Importing Models

Before virtually describing an assembly task, the parts of that task need to be modeled before importing it into the interface. FreeCAD[2], an open source 3D CAD modeler with surface and mesh export capability is used (see Figure **??**). Engineering design requires in-context parametric modeling to handle the complexities of dealing with industrial assembly tasks. A major issue is that a plurality of mainstream modeling software often uses its own proprietary format. Traditionally exported file formats include the STL [21] file format when dealing with geometric structure and 3D

[1]https://protege.stanford.edu/
[2]https://www.freecadweb.org/

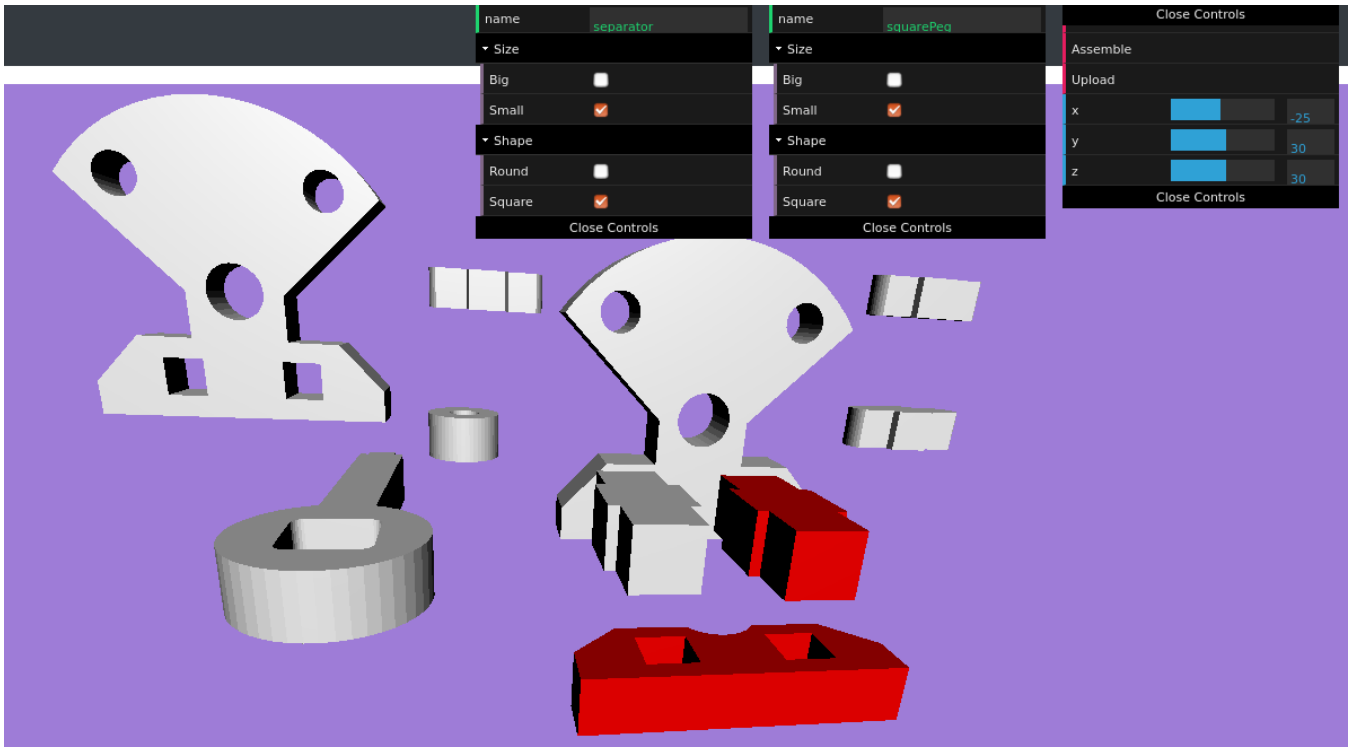

Fig. 3. Interaction in the VirtualAssembler. Each part can be moved around separately and the camera can be moved to increase visibility of the scene. Selecting one part will allow to specify individual properties. Once two parts have been selected they can be linked together to indicate that they should be assembled during the execution phase.

printing. The STEP file format is commonly used when sharing tolerances for physical assembly components. These file types grow in size beyond acceptable for client nodes as they are designed to store details on workstations.

The developed environment, denoted VirtualAssembler, uses a WebGL [22]-based graphical interface that requires the GLTF [23] file format. Therefore, assemblies need to be translated from the CAD software formats into GLTF which is a compact format that is easily processed by the GPU. The GLTF file format stores the geometry, scene, camera, keyframes, material and textures, unlike STL which represents a model through unit normals and vertices, in GLTF they are defined as arrays of primitive triangles and their attributes (e.g, colors, tex_coords, materials, etc.). This conversion requires a robust implementation as the model details can break off at sharp corners. Blender [24] is a free modeling application that has utilities to perform this conversion.

B. Interacting with Models

The VirtualAssembler makes use of Flask [25], a python-based micro web framework and Three.js [26] a javascript library based on WebGL, used to integrate 3D model visualization into browser applications. This interface provides two types of control schemes:

- A 3D graphical drag and drop system.
- Standard GUI.

Using the drag and drop system the user can select pairs of objects in the 3D view that are to be assembled together (see Figure 3). The selection procedure uses a ray caster to get intersection points on the ray from the camera to the pointer. The standard GUI allows the user to select parts from a list and access their various properties. Here, the user can define whether a part is a peg or contains holes and shafts. In addition, the user can also mate parts with others and determine their mating properties without graphical interaction. Once the user has specified as many steps in the assembly process as deemed necessary, the planner can be evoked to generate the entire action sequence.

C. Exporting Ontologies

All the information gathered from the VirtualAssembler is loaded into an ontological database. Once stored this data can be used for future assembly tasks together with the basic ontology file written using the Protege Editor [27]. Pieces of interest include:

- The parts involved in the assembly.
- Properties that could help the robot place or grasp.
- Links between parts.

The conversion is two-fold. First, on the Flask server, each part is converted into a JSON object together with the links between them that have been previously specified by the user. That JSON object is sent to the Resource Description Framework (RDF) reasoner where properties such as shape are taken into account to generate a complete specification than is then used as the basis for a plan. Once the ontology file is generated, it needs to be manually imported into the ontology server holding all the information. The server is an

Apache Jena Fuseki server [28], providing the SPARQL [29] protocols needed to manage the data. SPARQL is a query language for data within the RDF format, that same format used to store ontologies.

IV. INTERACTIVE PLANNING

We use NetworkX [30], a python library to manipulate graphs which are used to model simple temporal networks (STNs). STNs are directed graphs, where nodes (time variable) represent events corresponding to different phases of an assembly, and edges (time constraints) are temporal intervals between nodes. A solution is reached when all time variables have been assigned a completion time that satisfies all constraints in the netwwork. The interactive planner has two main components, the planner and the dispatcher. The planner is used to perform most of the computation work off-line, the design of which is based on the implementation described in [31]. The dispatcher reacts to temporal conflicts within the original plan to ensure the robot adapts to unexpected events.

A. Establishing a Complete Plan

First, the ontology needs to be translated into a graph that captures the temporal information of the task. Algorithm 1 shows the transition between the information contained in an ontology file and a Simple Temporal Problem, first step of the planning stack.

```
Data: Name of a skill
Result: Generate a simple Temporal Problem ready to
        be sent to the planner
begin
    nodes = []; assemblies = [];
    steps = retrieve_assembly_steps(skill);
    foreach step do
        assemblies.append(retrieve_links(step));
    end
    step_id = 0;
    while assemblies not empty do
        list_tasks = find_min_constrained(assemblies);
        assemblies = assemblies - list_tasks;
        foreach link in list_tasks do
            nodes.append(create_node(link));
        end
        create_edges(nodes);
    end
end
```

Algorithm 1: SPARQL queries are sent to the ontology server to retrieve the necessary information. The reasoner then generates the assembly nodes and uses constraints to deduce the edges between them.

Assemblies correspond to time points and parts involved in several assemblies will generate the time constraints as to what should be done first. The reasoner uses the information it has about the parts to deduce the order of actions.

B. Interacting with the Plan

At this point, the information seen on the screen is what the robot will expect to happen. It is very important in a collaborative environment that all parties know what other agents are about to do. The robot will announce the step it is about to undertake but humans do not have this requirement, with our current developments this would lead to an unreasonable cognitive load. Updated while the actions are taken, if the user decides to modify the plan it can do so using the VirtualAssembler. The agent responsible for a specific assembly step in the robot planning module can be changed.

C. Dispatching a plan

Once a plan has been compiled off-line it is ready to be dispatched. In previous works we had discussed policies to solve the task allocation problem [27]. At run time the robot is the master and will thus choose a valid set of task assignments giving its teammate a list of actions doable at the time while holding a similar list for itself. Each node of the graph corresponds to an assembly action command and contains the names of the peg and the part it should fit in.

In case the human undertakes an assembly step, the robot needs to update the plan accordingly. To send feedback we use the VirtualAssembler interface in which it is possible to specify that a step has been completed.

V. TASK EXECUTION

We use ROS and moveIt with the Panda arm, a 7 degrees of freedom robotic arm from Franka Emika. The teaching phase is illustrated with the Cranfield Benchmark assembly. It involves ten parts. Some of which are identical in shape and size, and need to be assembled in a specific order to correctly complete the task, while other steps can be completed independently of one another.

A. Cranfield Benchmark Assembly

The general rules of interest considered in the ontology (see Figure 4) to establish a plan are:

- Whether the part is a peg or contains holes,
- Properties of the parts, such as its size and its shape (or the shape of its holes),
- Links between the parts, indicating an assembly step.

Let us consider a subset of the Cranfield Assembly benchmark (see Figure 4). Assembly 1 and 2 are the insertions of the two square pegs into the front plate. Assembly 3 is the separator coming on top of them. Their properties can be annotated in the virtual assembler as shown in Figure 3.

The plan associated with this subset will then be composed of three nodes, one for each assembly. Assemblies 1 and 2 are independent and are thus not linked by an edge but each of them is linked to the Assembly 3 as they need to be inserted before the separator.

During the execution, the Assembly will initially be left out and the robot will only display the 2 first ones. According to the policy it will suggest to the human which agent should be responsible for which task.

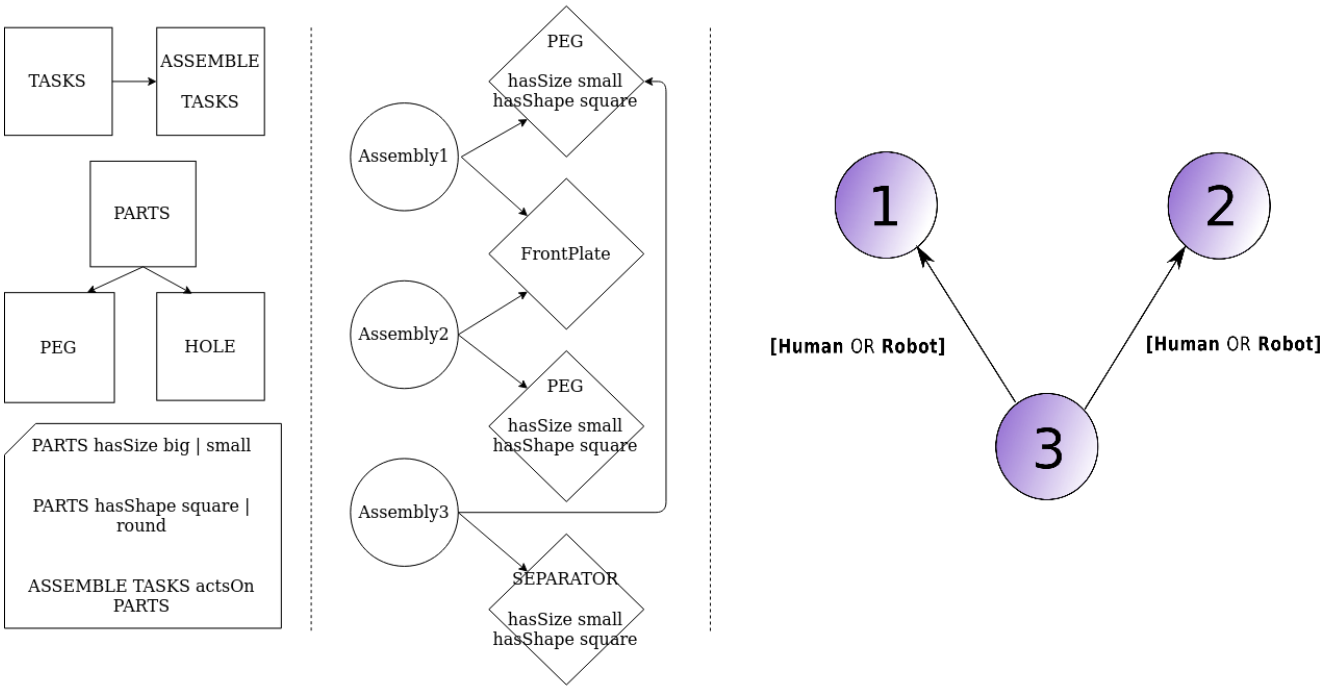

Fig. 4. Structure of the entities created when the virtually assembly is translated to an owl file. Left-most figures describes the rules defined in the main ontology and based on [27]. Middle figure depicts the individuals (i.e. instances) of those entities. Right figure depicts the Simple Temporal Model created from this extract of knowledge. The modelled Simple Temporal Problem, based on the extraction of ontology containing information about the Cranfield Assembly, can be explained as follows. Each node holds the information about an assembly to be performed. 1 and 2 correspond to inserting a square peg into the faceplate. 3 is putting the separator on top of those two pegs. Each edge represents a time constraint to synchronize tasks. At this stage they are composed of two time ranges, corresponding to human and robot performances respectively.

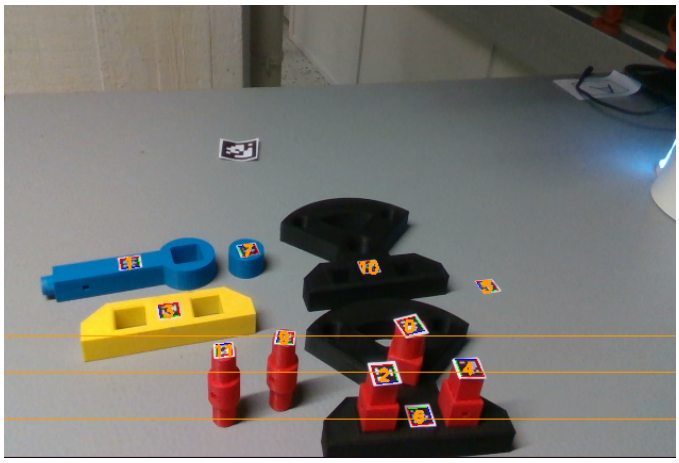

Fig. 5. Top: Our workspace where a human agent can perform a task together with the Panda arm from Franka Emika. Detection of those AprilTags through ROS.

Each node contains the type of instruction to be sent to the robot. In this case we only consider assembly tasks, as well as the list of parts involved in the step.

B. Visual Tracking

To increase the versatility of our approach we do not require parts in the assembly task to be placed in any specific location on the working surface. Therefore it is necessary to include a visual sensor to perceive the current state of the environment.

In order to facilitate real time tracking and identification of all relevant parts in the assemble task we placed AprilTags [32] of varying sizes on each part. Visual detection of an apriltag within the camera frame reveals the position and orientation of the part assigned the number tag (see Figure 5). This information can then used to update a digital twin contained within the planning scene of the robot.

VI. DISCUSSION

Semantic annotations and 3D CAD model manipulation. The system in its current form allows for 3D CAD model annotation by the user. While this enables to specify relevant task knowledge and part properties, in most cases these have to be specified by a person. The choice of what to include and what to ignore is difficult to answer, as this depends on the task and the functionality of the system. For example, in the case of the Cranfield assembly, hole positions in the faceplate were hard coded in the ontology. Alternatively, these can be extracted automatically from CAD data or detected by a vision system.

Creating a plan. Some parts of an assembly task are more complex as they belong to both categories (peg and hole) simultaneously. For example, the pendulum in the Cranfield benchmark behaves as a peg with the pendulum head it, the pendulum also has a hole so that it can be attached to the rest of the assembly. Completely automating the assembly plan can lead to errors due ambiguity of some parts and unknown

constraints. The interactive planner is thus an important feature that enables the user to specify constraints and resolve ambiguities prior to automating the rest of the assembly plan.

Reasoning about a plan. Defining every piece of knowledge the robot would have to know to be fully autonomous can become an intractable task. Moreover, even though defining new pieces of knowledge to integrate into the current state of the knowledge base is possible, the reasoning module is still fixed. Thus, it cannot take new types of tasks into account. For example the needs of an engine assembly would not be covered by the Cranfield assembly developments. This assembly task differs significantly and defining such complex task would lead to introducing new constraints that cannot be handled by the reasoner. Future work will therefore generalize our reasoner to accept modules defined externally.

VII. CONCLUSION

This work proposed a system to plan and execute shared assembly tasks between human and robot. Assembly tasks are extracted from a CAD model that describes relationships between individual parts and its properties, and is translated to an ontology for automated task plan reasoning. The interaction with the CAD model allows to add semantics to the parts for interactive planning. The capabilities of the system are illustrated with the Cranfield assembly benchmark which shows how the planner could use the information received after teaching.

ACKNOWLEDGMENT

International collaboration for this project was funded by the York University Vision Science to Application(VISTA) visiting Researcher Grant and the NSERC Canadian Robotics Network visiting Researcher Grant.

REFERENCES

[1] F. Tobe, "42 companies empowering robots and humans to work side-by-side," March 2017. [Online]. Available: https://robohub.org/42-companies-empowering-robots-and-humans-to-work-side-by-side/

[2] K. Collins, A. Palmer, and K. Rathmill, "The development of a european benchmark for the comparison of assembly robot programming systems," in *Robot technology and applications*. Springer, 1985, pp. 187–199.

[3] A. Pichler, S. C. Akkaladevi, M. Ikeda, M. Hofmann, M. Plasch, C. Wögerer, and G. Fritz, "Towards Shared Autonomy for Robotic Tasks in Manufacturing," *Procedia Manufacturing*, vol. 11, pp. 72–82, 2017.

[4] D. P. Losey, M. Li, J. Bohg, and D. Sadigh, "Learning from My Partner's Actions: Roles in Decentralized Robot Teams," *arXiv:1910.07613 [cs]*, Oct. 2019. [Online]. Available: http://arxiv.org/abs/1910.07613

[5] T. Taniguchi, T. Nagai, *et al.*, "Symbol emergence in robotics: a survey," *Advanced Robotics*, vol. 30, no. 11-12, pp. 706–728, 2016.

[6] K. Ramirez-Amaro, E. Dean-Leon, F. Bergner, and G. Cheng, "A Semantic-Based Method for Teaching Industrial Robots New Tasks," *KI - Künstliche Intelligenz*, vol. 33, pp. 117–122, June 2019.

[7] A. Roncone, O. Mangin, and B. Scassellati, "Transparent role assignment and task allocation in human robot collaboration," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore: IEEE, May 2017, pp. 1014–1021.

[8] M. Stenmark and J. Malec, "Knowledge-based instruction of manipulation tasks for industrial robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 56–67, June 2015.

[9] G. Sarthou, A. Clodic, and R. Alami, "Ontologenius: A long-term semantic memory for robotic agents," in *IEEE International Conference on Robot Human Interactive Communication (IEEE RO-MAN)*, 2019, p. 9.

[10] T. R. Savarimuthu, A. G. Buch, C. Schlette, W. Nils, *et al.*, "Teaching a robot the semantics of assembly tasks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 5, pp. 670–692, 2018.

[11] W. Wang, R. Li, Y. Chen, Z. M. Diekel, and Y. Jia, "Facilitating Human-Robot Collaborative Tasks by Teaching-Learning-Collaboration From Human Demonstrations," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 1–14, 2018.

[12] M. H. Arbo, Y. Pane, E. Aertbelien, and W. Deere, "A System Architecture for Constraint-Based Robotic Assembly with CAD Information," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. Munich: IEEE, Aug. 2018, pp. 690–696.

[13] G. Michalos, J. Spiliotopoulos, S. Makris, and G. Chryssolouris, "A method for planning human robot shared tasks," *CIRP Journal of Manufacturing Science and Technology*, vol. 22, pp. 76–90, Aug. 2018.

[14] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning Robotic Assembly from CAD," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 3524–3531.

[15] L. Johannsmeier and S. Haddadin, "A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 41–48, Jan. 2017.

[16] S. Bogomolov, D. Magazzeni, A. Podelski, and M. Wehrle, "Planning as model checking in hybrid domains," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI'14. AAAI Press, 2014, pp. 2228–2234.

[17] J. A. Shah, "Fluid coordination of human-robot teams," Ph.D. dissertation, Massachusetts Institute of Technology, 2011.

[18] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1, pp. 61 – 95, 1991.

[19] C. Combi, R. Posenato, L. Vigan, and M. Zavatteri, "Conditional Simple Temporal Networks with Uncertainty and Resources," *Journal of Artificial Intelligence Research*, vol. 64, Apr. 2019.

[20] B. Hayes and B. Scassellati, "Autonomously constructing hierarchical task networks for planning and human-robot collaboration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden: IEEE, May 2016, pp. 5469–5476.

[21] L. Roscoe *et al.*, "Stereolithography interface specification," *America-3D Systems Inc*, vol. 27, 1988.

[22] C. Marrin, "Webgl specification," *Khronos WebGL Working Group*, vol. 47, p. 48, 2011.

[23] KhronosGroup, "Khronosgroup/gltf," Oct 2019. [Online]. Available: https://github.com/KhronosGroup/glTF/tree/master/specification/2.0

[24] B. Foundation, "Home of the blender project - free and open 3d creation software." [Online]. Available: https://www.blender.org/

[25] "Welcome to flask." [Online]. Available: https://flask.palletsprojects.com/

[26] "Three.js." [Online]. Available: https://threejs.org/

[27] A. Angleraud, Q. Houbre, and R. Pieters, "Teaching semantics and skills for human-robot collaboration," *Paladyn, Journal of Behavioral Robotics*, vol. 10, no. 1, pp. 318–329, Jan. 2019.

[28] A. Jena, "A free and open source java framework for building semantic web and linked data applications," *Available online: jena. apache. org/(accessed on 28 April 2015)*, 2015.

[29] A. Seaborne, G. Manjunath, C. Bizer, J. Breslin, S. Das, I. Davis, S. Harris, K. Idehen, O. Corby, K. Kjernsmo, *et al.*, "Sparql/update: A language for updating rdf graphs," *W3c member submission*, vol. 15, 2008.

[30] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," in *7th Python in Science Conference (SciPy2008)*. aroquaux, G., Vaught, T., Millman, J. (Eds.), 2008, pp. 11–15.

[31] A. Angleraud, Q. Houbre, M. Netzev, and R. Pieters, "Cognitive Semantics For Dynamic Planning In Human-Robot Teams," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. Vancouver, BC, Canada: IEEE, Aug. 2019, pp. 942–947.

[32] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2016, pp. 4193–4198.