Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 217 (2023) 306–315

Procedia Computer Science

www.elsevier.com/locate/procedia

4th International Conference on Industry 4.0 and Smart Manufacturing

Artificial Intelligence Task Planning of Cooperating Low-Cost Mobile Manipulators: A Case Study on a Fully Autonomous Manufacturing Application

Stefan-Octavian Bezrucav[a,*], Nils Mandischer[a], Burkhard Corves[a]

[a]*Institute of Mechanism Theory, Machine Dynamics and Robotics, RWTH Aachen University, Eilfschornsteinstr. 18, 52064 Aachen, Germany*

Abstract

Through the highly innovative processes introduced by the movement Industry 4.0, application of fully autonomous working processes is ready to be integrated into real-world manufacturing applications. While many tasks still require the human to handle particular parts or perform certain steps in the production chain, partial assembly may be performed by cooperating mobile manipulators in the near future. However, many smaller companies already fail to adapt to the new trends, as Industry 4.0, in spite of its beneficial aspects, is costly to implement, particularly in still low-automated sectors. In this work, we show how artificial intelligence task planning may be integrated with computer vision systems to achieve a fully autonomous manufacturing process. We also show how low-cost robots are adapted to process demands on diverse system levels.

Keywords: task planning; computer vision; autonomous manufacturing; artificial intelligence; screw detection; low-cost robotics

1. Introduction

Industry 4.0 targets the full and flexible automation of logistics and manufacturing task. In such tasks, mobile manipulators play a major role. However, cooperating robots, whether with each other or with humans, are not yet fully integrated into Industry 4.0 processes. Automating a manufacturing tasks requires the integration and interaction of many systems: sensors, machines, logistics station, and robots alike. Hence, the overall complexity of automating such tasks is demanding on hardware, software, and costs of implementation, particularly for small and medium-sized enterprises (SMEs) and companies in still low-automated sectors. One example of such low-automated process is the assembly of rotary tables at the Goizper Group, which was handled during the SHAREWORK EU-project (https://sharework-project.eu/). At the beginning of the project, all tasks were performed exclusively by humans.

* Corresponding author. Tel.: +492418099789 ; fax: +492418092263.
E-mail address: bezrucav@igmr.rwth-aachen.de

1877-0509
10.1016/j.procs.2022.12.226

SHAREWORK solutions enable in-between Human-Robot Collaboration (HRC) in parts handling. In these processes, cooperating mobile manipulators support the humans.

In SHAREWORK project, a software suite of 15 modules is developed that enables HRC in industrial scenarios without requiring the delimitation by spatially dividing safety measures, such as fences. These modules range from environment cognition [1], to automated task and trajectory planning [2], safety, and sociological and ethical factors. This work focuses on two of the SHAREWORK modules: environment cognition and automated task planning. The environment cognition module supervises the work process. In this paper, a camera system detects (a) screws that have been inserted into the rotary table for further assembly steps and (b) if the correct parts have been delivered to the logistics stations (by the human or other agents). The automated task planning acts as a high-level control loop, determining and managing the actions of the autonomous and human agents.

In this work, we introduce a fully integrated automation solution for a simple, but comprehensive, manufacturing process. The focus is not set on the development of new environment cognition or task planning methodologies. State-of-the-art algorithms are implemented and deployed in the proposed system. The novelty comes from the integration of these modules in a high-level, easily-configurable control system that enables an autonomous, highly-flexible application. In this application, cooperating mobile manipulators pick screws from logistics stations and place them into a rotary table, while dealing with noisy sensors or unpredictable human behaviours. Afterwards, the screws are fastened by a serial manipulator mounted directly to the work station (which is not covered in this paper). The partial process step of item delivery will still be performed by the human once SHAREWORK ends. By applying our work, the process may be fully automated in the near future. To further cope with the issue of Industry 4.0 being expensive, we decide to integrate the process only using low-cost technologies. This includes sensors and robotic hardware. As a consequence, all system components are subject to noise, for which strategies will be proposed in the course of this paper.

The paper is structured as follows: Firstly, we discuss related work (Section 2) before describing the deployed methodologies for automated, artificial intelligence task planning (Section 3) and screw detection (Section 4). Secondly, we verify the feasibility of the combined methodology in a lab scenario (Section 5) and, finally, conclude with an outlook on future research and application of the proposed methods in industrial applications (Section 6).

2. Related Work

Robotic systems have been enhanced in the last years with even more complex cognitive capabilities [3] that allow their tight integration in processes currently executed exclusively by humans [4]. Theses cognitive enhancements usually come in form of software modules that simplify customization and integration. Two of these modules are particularly relevant for this work, namely task planning and items detection. This section discusses related work for these modules.

2.1. Task Planning

In industrial applications, task scheduling and task planning modules sustain production processes by dynamically allocating tasks to the agents depending on the actual state of the system. In the works proposed by Cesta et al. [5] and Umbrico et al. [6], knowledge representation and reasoning is used by an Artificial Intelligence (AI) task planner to dynamically distribute the tasks to a team containing a human and a serial robot. Another AI planning paradigm is used in [7] to compute the tasks for a mixed team of one human and one mobile manipulator. The two agents share their working areas as part of an assembly scenario. The robot agent from this work has more elaborate skills (e.g., can navigate and execute trajectories with its arm) that increase the complexity of the tasks computation and allocation processes. Fiasch et al. [8] propose an different job allocation method. Their algorithm takes the capabilities, skills, knowledge, and the preferences of the worker into account, and solves a non-linear optimization problem to determine the jobs allocation. Other task allocation methods for industrial logistics and production environments that also consider agents' skills and capabilities are proposed in Pedersen and Krüger [9] and Ranz, Hummel, and Sihn [10].

2.2. Screw Detection

Computer vision is a key requirement to Industry 4.0. For the proposed scenario, artificial scene understanding is utilized in the form of screw detection. Such methods are usually split into two sub-problems: Firstly, segment screw candidates, and, secondly, classify the screw candidates into screw or non-screw classes. Ramana, Choi, and Cha [11] and Wegener et al. [12] both propose classifiers based on adaptive boosting (AdaBoost) to detect screws on 2D images. Cruz-Ramirez et al. [13] propose a multi template matching classifier. However, their approach fails to adapt to new types of screws. Li, Wei, and Xing [14] apply a similar method for feature detection and embed it into a Support Vector Machine (SVM) for classification. Tellaeche, Maurtua, and Ibarguren [15] improve on the template matching by using 3D CAD data for template generation. While template matching and SVMs are common for screw detection, tree-based methods are not applied outside the AdaBoost approaches proposed by [11] and [12]. While the priory mentioned methods use conventional learning approaches, Yildiz and Wörgötter [16] apply the Hough Circle Transform to find screw candidates and classify them using neural networks. Martinez, Ahmad, and Al-Hussein [17] improve on the method by applying an ellipse fitting algorithm to compensate for different view angles. However, while neural networks achieve good results, they cannot be adjusted by untrained personnel. In case of transporting digitization to low-automated sectors, this is a hurdle not solvable in industrial practice. Therefore, we focus on more traditional learning methods.

3. Automated Task Planning

This section briefly introduces automated planning methods and focuses on the planning model formulated for the targeted scenario.

3.1. Background

Automated task planning or AI task planning methods compute the actions that must be executed in a system to bring this system to a desired goal state. In contrast to scheduling methods, which just arrange a set of predefined actions to determine a plan, AI task planning methods perform two steps. In a first step, these methods select and instantiate actions from a set of abstractly defined ones. The selected and instantiated actions are those that can bring the considered system from an initial state to a given goal state. The solving process is a search process that traverses several states from a planning state space. The search is guided by heuristics functions that determine the effort required to reach a goal state from one of the expanded states. Further, each state of the state-space encodes several pieces of information about the targeted application, while each transition between two states correspond to the execution of one action. In this setup, an action can be executed in a state only if a set of requirements are met. For example, a *grasp* action can be executed only when the agent is at a location where the object to be grasped is present. In a second step, AI task planning methods optimize the plan with respect to a set of criteria (e.g., time). These methods adapt the start times and order of the planned actions, without violating the dependencies between them, to obtain a plan with minimal makespan (execution time). As input, AI task planning methods require only the abstractly-defined actions, an initial planning state, and a set of goals that must hold in a goal state. [18]

Considering these characteristics, AI task planning methods can be used as high-level control strategies in dynamic scenarios with robots and humans. These methods are able to generate new plans for many planning instances, for example, when new orders arrive. For each new planning instance, only the initial and the goal states must be re-set, while the remaining definitions and the planning process itself must not be further adapted. Two AI task planning approaches, namely classical planning and temporal planning, were already successfully deployed in several of such dynamic applications [7, 19, 20]. The temporal planning approach is a planning method that works with durative actions $a_i \in A$. A durative action a_i is an action with a set of conditions $cond(a_i)$, a set of effects $eff(a_i)$, and a specific duration $\Delta(a_i)$. Action a_i can be executed in a state s only if its conditions hold in that state. In addition, by executing action a_i, its effects are applied to the system which is transformed to a new state.

As mentioned above, each action a_i is defined in a generic way. For example, a *navigate* action can be executed by an *agent*, from a *pose_from*, to a *pose_to*. The planning process implies, among others, the instantiation of these

actions to specific values (e.g., *navigate robot pose1 pose2*). The instantiation process is correlated with the ordering of the actions, such that all dependencies between actions conditions and effects hold. The result is a plan

$$\pi =< a_0, \ldots, a_n > \tag{1}$$

that contains n actions. By executing all actions of π, the initial state of the system s_0 is repetitively changed to intermediate states s, as described by the instantiated effects of these actions, until a goal state s_g is reached where the defined goals g hold.

Planning Domain Definition Language (PDDL) is the standard language automated planning problems are formulated in [21]. Each planning problem is described in a PDDL domain and a PDDL problem file. The PDDL domain file contains the definition of the types, the predicates, and the generic definition of the actions. During the planning process, objects of the defined types are created. The predicates are boolean functions with any number of parameters (of the previously defined types). Each action definition contains a set of parameters, the *at start*, *at end*, and *overall* conditions, as well as the *at start* and *at end* effects. The complete formulation of a planning problem further requires the initialization of the objects and the setup of the initial state and the goals. The elements that are not described in the PDDL domain file are integrated in the PDDL problem file. With these two files, temporal planners such as optic [22] can be deployed to compute a plan.

3.2. Planning Model

The planning problems for the targeted use-case are described with a PDDL domain file containing three actions: *navigate*, *grasp*, and *discard*.

```
1  (:durative-action navigate
2  :parameters (?agent - agent ?from ?to - agentpose)
3  :duration (= ?duration 10)
4  :condition  (and (at start (at ?agent ?from))
5  (at start (not_acting ?agent))
6  (at start (free ?to))
7  (at start (navigate_allowed ?agent)))
8  :effect  (and  (at start (not (at ?agent ?from)))
9  (at start (not (free ?to)))
10 (at start (free ?from))
11 (at start (not (not_acting ?agent)))
12 (at start (not (navigate_allowed ?agent)))
13 (at end (at ?agent ?to))
14 (at end (not_acting ?agent))))
```

Fig. 1. Excerpt of the PDDL domain file with the definition of the *navigate* action.

Figure 1 depicts the PDDL formulation of the *navigate* action. Such an action can be introduced in a plan if the agent is at the *from* pose (line 4), the *to* pose is free (line 6), the agent is not acting (line 5), and it may *navigate* (line 7). The *not_acting* and *navigate_allowed* predicates impose a sequential execution of actions for each agent and the execution of another action between each two *navigate* actions, respectively. A more detailed explanation of these two predicates is given in [23]. Once the execution of a navigate action is started, the *at start* effects are set: the agent is not at the start pose anymore (line 8), the *from* pose is marked as free (line 10), and the *to* pose as not free anymore (line 9). Further, the *not_acting* predicate is negated (line 11). This implies that the agent is acting during the execution, fact that is deleted at the end of the execution (line 13). By the end of the execution, the agent should have reached the *to* pose (line 12) and it is not allowed to immediately execute a further *navigate* action (line 14).

The *grasp* and *discard* action are implemented in a similar way. They start with a set of parameters and a duration value. They also contain a list of conditions that must hold before the action can be planed. For example, for a *grasp* action, the agent and the thing to be grasped must be at the same location and the agent must have its gripper free. If

the action is planned, a set of effects are set. The most relevant ones model the swapping of the positions for the thing, between the position on a table and the position in the gripper of the agent.

```
1  (:objects
2  robot_one - robot
3  robot_one_thingpose1 - thingpose
4  in_pose table_pose_left table_pose_right - agentpose
5  in_pose_thingpose1 ... - thingpose
6  table_pose_thingpose1 table_pose_thingpose2 ... - thingpose
7  screw1 screw2 ... - element
8  nothing - no_thing)
9  (:init
10 (at robot_one_thingpose1 robot_one)
11 (at in_pose_thingpose1 in_pose) ...
12 (at table_pose_thingpose1 table_pose_left) ...
13 (thing_moveable screw1) ...
14 (thing_placeable screw1 table_pose_left) ...
15 (thing_for_agent screw1 robot_one) ...
16 (at robot_one table_pose_left)
17 (free in_pose) ...
18 (at screw1 in_pose_thingpose1) ...
19 (at nothing robot_one_thingpose1) ...)
20 (:goal (and
21 (process_step_done screw1 step1) ... )
```

Fig. 2. Excerpt of the PDDL problem file with the definition of objects, the initial state, and the goals.

Figure 2 depicts the PDDL problem file for the targeted scenario. First, one robot, agentposes, and thingposes are initialized (lines 2-6). Afterwards, five screws and the nothing construct are created (lines 7-8). With the defined objects and the corresponding predicates from the domain file, the initial state s_0 is set up. First, fixed relations between the thingposes and the agents and agentposes are described (lines 10-12). Afterwards, the characteristics of the screws are set up (lines 13-15), before the actual distribution of the agents and of the screws in the world is presented (lines 16-19). Last, the goals are defined (line 21). The plan generated for the given planning problem is depicted in Figure 3.

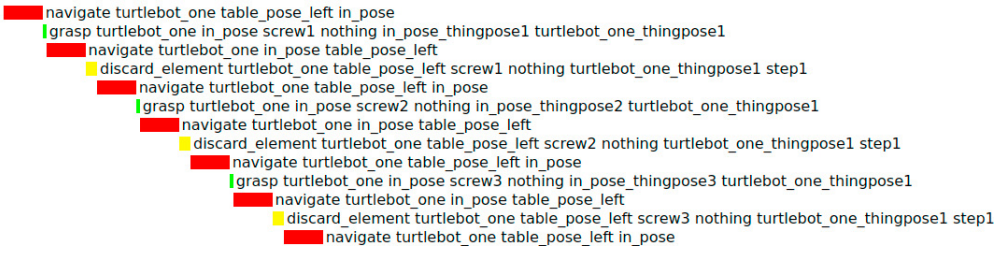

Fig. 3. Section of plan π obtained for the Goizper planning problem.

3.3. AI Task Planning Framework

The generated actions must be executed in the real scenario. Hence, the planning process is integrated in an extended version of the ROSPlan framework [7] (based on [24]) that acts as a high-level control loop. ROSPlan contains a knowledge base in which the values of the predicates are saved and which tracks their evolution as they change during the actions' execution. ROSPlan also integrates four other modules; the most important ones being those that generate a planning problem from the information saved in the knowledge base, call a planner, parse the obtained plan, and dispatch the planned actions.

Each planned action (e.g., from the plan depicted in Figure 3) is initially described in an abstract way. Therefore, it must be translated to another data structure that implements its sub-steps and that is connected to the modules that allow the execution in the real world. The new data structure is a finite state machine (FSM). For each such action, an executor finite state machine (EFSM) is implemented. The *discard* action has the most relevant EFSM as it integrates calls to the modules of the agent, the environments, and the automated screw detection.

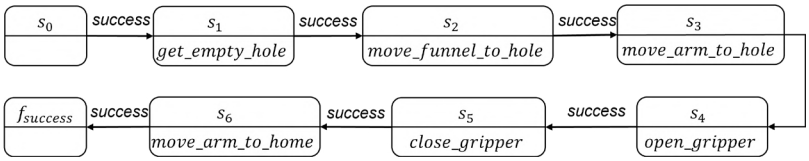

Fig. 4. Executor finite state machine implementing the *grasp* action.

Figure 4 presents the *discard*-EFSM. It contains 8 states and implies the following sequential execution:

1. Call to the screw detection module to get an empty hole where the screw carried by the robot should be placed,
2. Call to the funnel system to move the funnel over the corresponding hole,
3. Call to the robot arm to execute a trajectory to a fixed pose over the funnel,
4. Call to the robot gripper to open the gripper and release the screw,
5. Call to the robot gripper to close the gripper,
6. Call to the arm to retract to its home pose.

All calls are connected to Robot Operating System (ROS) service servers that are running on the robot or on distributed systems in the environment (e.g., the funnel system). Similar EFSMs are modelled also for the navigate and grasp actions.

The planning problem integrated in the extended ROSPlan framework and the description of the EFSMs for all planning actions enable the autonomous execution of the screw fitting processes from the real use-case. Further, independent of the number of screws to be inserted and of the screws that may be inserted by a human worker, the system can adapt autonomously and finalize the assembly process.

4. Screw Detection

To extend the state of research, we propose to use a tree-based learning approach for distinguishing holes and screws. In the described application, it is more important to detect empty holes than to distinguish screws from background. The two classes problem has not been covered profoundly in literature. In recent research, we observed good generalization of Random Forest classifiers [25] for applications with low training sample sizes. To allow lenient application of the methodology for untrained personnel, we propose following pipeline: Firstly, images are captured with a camera mounted parallel to the rotary table. The images are then segmented into screw candidates and manually sorted into screw and hole classes for classifier training. The segmentation of images is performed autonomous and the user is only presented with the cropped images. Lastly, the classifier is trained and may be applied directly to the manufacturing task.

4.1. Method

For segmenting screw candidates, we exploit some geometrical features of the rotary table that is to be assembled. The screws are located on a ring segment of a concentric and circular counter-plate (rotary disk). Hence, we first define a region of interest (ROI) defined by a ring segment of particular width, relative to the outer contour of the rotary disk. As the camera is placed over the center of gravity of the rotary disk, Hough Circle Transform (HCT) [26] may be well utilized to get the outer contour of the assembly. From the outer ring, the ROI is moved inwards based on the geometric measures. Hence, the ROI is embodied by a thin ring element around the screws and holes. Afterwards, HCT is applied onto the ring segment to segment the screw candidates. The screw candidates are then evaluated with a

Random Forest classier. To make the classifier better understandable for the human operator, we choose to only apply five features that are directly derived from real-world observation:

1. Brightness intensity of the candidate's center area (25% diameter); this is the reflection of the workshop lights in the screw head
2. Average brightness intensity of candidate area
3. Brightness intensity of outer ring (50-100% diameter); this is the reflection of the workshop lights on the countersunk shoulder in the hole
4. Number of local brightness maxima
5. Number of pixels of brighter intensity than average on the candidates main axis.

An exemplary detection result after training is depicted in Figure 5.

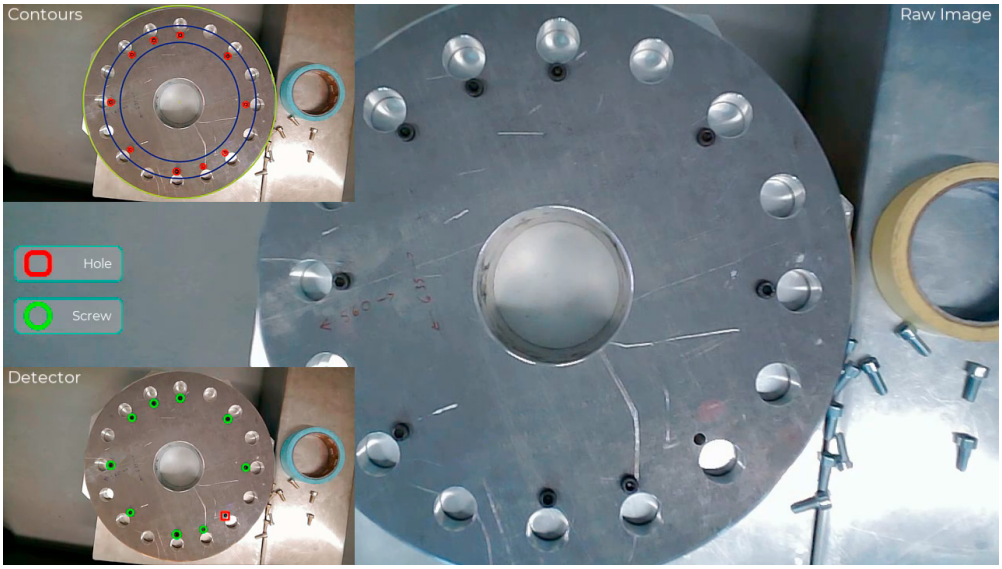

Fig. 5. Screw detection module with raw image (right), detected outer contour (green ring) and ROI (blue ring segment), and detected screws (bottom-left; screws: green, holes: red).

4.2. User Interface and Application with Noise

To train the classifier, the user is guided through a command line-based user interface. First, the user needs to take images with the camera in the desired setup. Afterwards, the images are cropped and presented to the user. This process is based on the same HCT segmentation approach as discussed in the prior section. The user then decides if the cropped image shows a screw, a hole, or background. The cropped images are then sorted internally and stored for training. Once the user has classified all training samples, the classifier is automatically trained.

A machine learning classifier on real data is never fully accurate as environmental conditions and placing of parts may change over the course of runtime. Therefore, additional measures have to be taken to deal with the uncertainty in (a) the localization of screw candidates and (b) the classification. First, all holes' known positions are referenced in the middle point of the rotary disk, which is supplied with a coordinate frame relative to the main rotational axis. By this, all potential screw positions are fully known independently of the rotary disk's orientation. When receiving a screw candidate, they are matched onto the known positions using a spatial threshold. After classification, the class decision is stored as vote on the specific location. A maximum of ten votes is stored per location and old votes are discarded if the maximum is exceeded. Once the task planner requests empty holes, the votes are consulted. If the certainty of the votes towards the hole class exceeds 60%, the candidate is declared a hole and the ID is returned to the task planner.

We validate the classifier with a total of 1000 cropped training samples in the setup discussed in Section 5. The end-to-end classification approach reaches an accuracy (correctly classified samples over all samples) of 87.3%. By this, the threshold of 60% for determining the vote is valid, as it is lower than the expected certainty of the vote. Note that end-to-end classification also incorporates errors in the HCT and localization of screw candidates.

5. Verification in a real industrial application with low-cost robots

Figure 6 depicts the adapted SHAREWORK Goizper-scenario. The test scenario is a recreated industrial environment delimited by walls from the rest of the laboratory. Inside the working area, different tables and locations are given. The ones relevant for this scenario are the input table where the screws are gathered and the table on which the rotatory table is located. The rotatory table is enhanced with a funnel system that compensates the errors in the localization and navigation of the low-cost mobile base and enables the exact placing of screws in the corresponding holes. Note that without further measures a precision of 1mm would be required to perform the insertion of screws in the corresponding holes. Above the rotatory table, the camera system (https://www.stereolabs.com/zed-2/) used to recognize the screws is installed. As agents, one mobile manipulator and a human are involved.

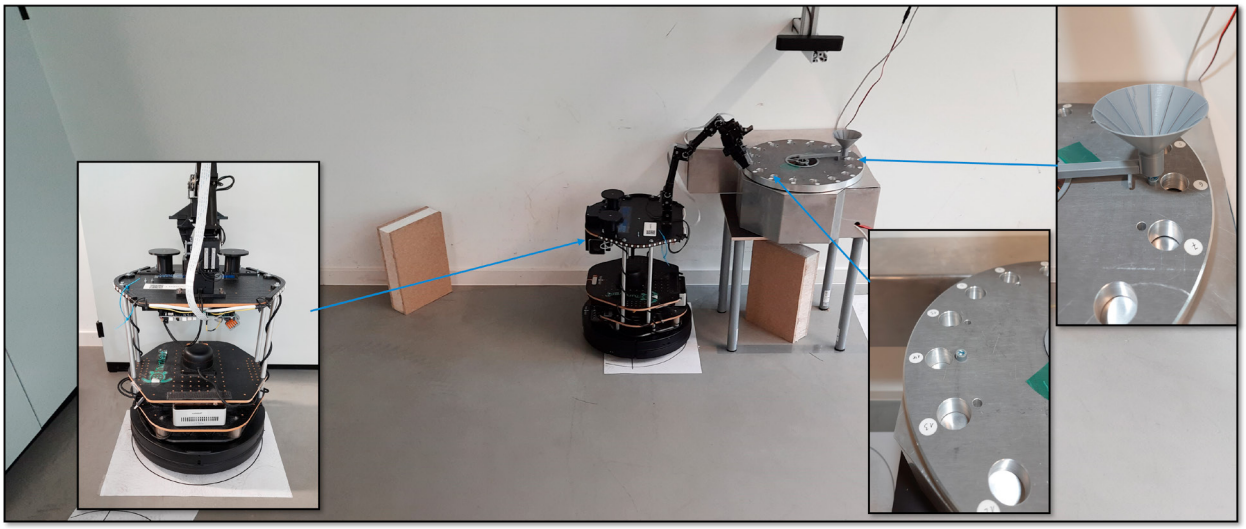

Fig. 6. Overview of the scenario with the involved mobile manipulator, the rotatory table with screws, and the funnel system.

At start, the robot is located anywhere in the environment and no screws are placed into the rotatory table. In the goal state, two screws must be inserted into holes from the right and two screws from the left side of the rotatory table. The automated task planning module computes a plan with *navigate*, *grasp*, and *discard* actions for the mobile manipulator. The *navigate* actions are carried out between the *in* pose and two other poses: left and right of the rotatory table. The *grasp* action is executed at the *in* pose where the human is required to hand over a screw to the mobile manipulator. The *discard* action is carried out at the rotatory table and involves the robot, the camera system, and the funnel system. During execution of the plan, the human may intervene at any time and place new screws into any hole of the rotary table.

The combined system containing the task planning framework and the screw detection module is integrated and validated. The system is able to automatically recognize the situation and adapt the plan execution accordingly. We explicitly tested and verified:

- Execution of initial plan without external disruptions,
- Disruption by the human walking in the delimited area,
- Disruption by the human not handing over desired items,
- Disruption by changing the insertion state of screws in the rotary table.

All scenarios are executed successfully. A demonstration of the real experiment is available as a video (*https://www.youtube.com/watch?v=oIxrqpY-Ku8*).

6. Conclusion and Future Work

In this paper, we show how a typical manufacturing task may be fully automated with low-cost robots in a combined effort of task planning, computer vision, and task execution. First, we show how the task of assembling a rotary table at the Goizper Group is modelled using PDDL. Then, we embed the methodology into an AI planning framework that is enhanced with an AI screw-detection module. The screw detection module is based on Random Forest classification of holes and screws. Finally, we show how the model and system components are composed into a fully autonomous demonstration of the use-case.

The deployment of low-cost robotics reduces the financial effort for small and medium-size companies that want to automatize processes. However, cheaper software and hardware solutions imply further integration efforts, for example, when high-positioning accuracy is required and the deployed sensors are too noisy. These integration efforts are related to new hardware components or new software modules. In our case, the funnel system had to also be integrated in the use-case, because the mobile base of the agent is not able to achieve the required positioning accuracy. Robotic agents with more performant hardware and software might reduce this integration effort. Independent of the types of robots involved in the targeted application and their software and hardware capabilities, the proposed framework can still be deployed. Our framework combining task planning and environment cognition must only be configured and finely-tuned when used with different robots or in new applications. No new modelling from scratch is required. The adaptability and the flexibility of the framework are the most important advantages that characterizes our novel solution.

In future integration efforts, several SHAREWORK parts of the system will be integrated at the Goizper Group's facilities. During integration, we will collect data to analyze further improvements of the framework and underlying methodologies. One possible improvement is the extension towards more dynamic and non-deterministic tasks that will require an additional management of noise statistics.

Acknowledgements

This work has been partially funded by the European Union's Horizon 2020 project SHAREWORK (grant agreement No. 820807). In addition, we like to thank the Goizper Group and STAM S.p.A. for their support with integrating our methodologies in the use-case demonstrator at Goizper in Antzuola.

References

[1] Mandischer, Nils, Huhn, Tobias, Huesing, Mathias and Corves, Burkhard (2021) "Efficient and consumer-centered item detection and classification with a multicamera network at high ranges", in *Sensors*, vol. **21**, no. 14, 2021.

[2] Faroni, Marco, Beschi, Manuel, Ghidini, Stefano, Pedrocchi, Nicola, Umbrico, Alessandro, Orlandini, Andrea and Cesta, Amedeo (2020) "A Layered Control Approach to Human-Aware Task and Motion Planning for Human-Robot Collaboration", in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 1204-1210, doi: 10.1109/RO-MAN47096.2020.9223483

[3] Ingrand, Félix, Ghallab, Malik (2017), "Deliberation for autonomous robots: A survey", in *Artificial Intelligence* **247**: 10–44. DOI: 10.1016/j.artint.2014.11.003

[4] Wang, Lihui, Liu, Sichao, Liu, Hongyi, and Wang, Xi Vincent, (2020) "Overview of Human-Robot Collaboration in Manufacturing", In Wang, L., Majstorovic, V., Mourtzis, D., Carpanzano, E., Moroni, G., Galantucci, L. (eds) *Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing*, Lecture Notes in Mechanical Engineering. Springer, Cham. https://doi.org/10.1007/978-3-030-46212-3_2

[5] Cesta, Amedeo, Orlandini, Andrea, Bernardi, Giulio, Umbrico, Alessandro, "Towards a planning-based framework for symbiotic human-robot collaboration", in *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8, doi: 10.1109/ETFA.2016.7733585

[6] Umbrico, Alessandro, Orlandini, Andrea, Cesta, Amedeo, Koukas, Spyros, Zalonis, Andreas, Fourtakas, Nick, Andronas, Dionisis, Apostolopoulos, George, Makris, Sotiris (2021) "Towards user-awareness in human-robot collaboration for future cyber-physical systems", in *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1—8.

[7] Bezrucav, Stefan-Octavian, and Corves, Burkhard (2020), "Improved AI Planning for Cooperating Teams of Humans and Robots", in Michael Cashmore, Andrea Orlandini, Alberto Finzi (Eds.): *Workshop on Planning and Robotics (PlanRob) at International Conference on Automated Planning*. International Conference on Automated Planning and Scheduling (ICAPS).

[8] Fiasché, Maurizio, Pinzone, Marta, Fantini, Paola, Alexandru, Ana and Taisch, Marco (2016) "Human-centric factories 4.0: A mathematical model for job allocation", in *IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, 1–4, doi: 10.1109/RTSI.2016.7740613

[9] Pedersen, Mikkel Rath, Krüger, Volker (2015) "Automated Planning of Industrial Logistics on a Skill-equipped Robot", in *Workshop on Task Planning for Intelligent Robots in Service and Manufacturing, IEEE International Conference on Intelligent Robots and Systems. Proceedings* Hamburg, 2015.

[10] Ranz, Fabian, Hummel, Vera, Sihn, Wilfried (2017) "Capability-based Task Allocation in Human-robot Collaboration", in *Procedia Manufacturing*, **9**:182–189, doi: 10.1016/j.promfg.2017.04.011.

[11] Ramana, Lovedeep, Choi, Wooram, Cha, Young-Jin (2019) "Fully automated vision-based loosened bolt detection using the Viola-–Jones algorithm", in *Structural Health Monitoring*, **18**:422–434, doi: 10.1177/1475921718757459.

[12] Wegener, Kathrin, Chen, Wei Hua, Dietrich, Franz, Dröder, Klaus, Kara, Sami (2015) "Robot Assisted Disassembly for the Recycling of Electric Vehicle Batteries", in *Procedia CIRP*, **29**:716–721, doi: 10.1016/j.procir.2015.02.051.

[13] Cruz-Ramirez, S. R., Mae, Yasushi, Takubo, Tomohito, Arai, Tatsuo (2008) "Detection of screws on metal-ceiling structures for dismantling tasks in buildings", in *International Conference on Intelligent Robots and Systems*, IEEE, pp. 4123–4129, doi: 10.1109/IROS.2008.4650975.

[14] Li, Caiqin, Wei, Zhenzhong, Xing, Jing (2016) "Online inspection system for the automatic detection of bolt defects on a freight train", in *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, **230**:1213–1226, doi: 10.1177/0954409715588119.

[15] Tellaeche, Alberto, Maurtua, Inaki, Ibarguren, Aitor (2016) "Use of machine vision in collaborative robotics: An industrial case", in *21st International Conference on Emerging Technologies and Factory Automation*, IEEE, pp. 1–6, doi: 10.1109/ETFA.2016.7733689.

[16] Yildiz, EErenus, Wörgötter, Florentin (2019) "DCNN-Based Screw Detection for Automated Disassembly Processes", in *International Conference on Signal-Image Technology & Internet-Based Systems*, IEEE, pp. 187–192, doi: 10.1109/SITIS.2019.00040.

[17] Martinez, Pablo, Ahmad, Rafiq, Al-Hussein, Mohamed (2019) "Real-time visual detection and correction of automatic screw operations in dimpled lightgauge steel framing with pre-drilled pilot holes", in *Procedia Manufacturings*, **34**:798-–803, doi: 10.1016/j.promfg.2019.06.204.

[18] Ghallab, Malik, Nau, Dana S. and Traverso, Paolo (2016) "Automated planning and acting", Cambridge University Press, New York.

[19] Cashmore, Michael, Coles, Andrew, Cserna, Bence, Karpas, Erez, Magazzeni, Daniele; Ruml, Wheeler (2019) "Replanning for Situated Robots". In J. Benton, Nir Lipovetzky, Eva Onaindia, David E. Smith, Siddharth Srivastava (Eds.): *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling*. ICAPS. Berkeley, California USA, 11–15 July 2019: AAAI Press, pp. 665–673.

[20] Buksz, Dorian, Cashmore, Michael, Krarup, Benjamin, Magazzeni, Daniele, and Ridder, Bram (2018) "Strategic-Tactical Planning for Autonomous Underwater Vehicles over Long Horizons" in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, 01.10.2018 - 05.10.2018: IEEE, pp. 3565–3572.

[21] Fox, Maria and Long, Derek (2003), "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains", in *Journal of Artificial Intelligence Research (JAIR)*, **20**: 61–124.

[22] Benton, J., Coles, Amanda, Coles, Andrew (2012), "Temporal Planning with Preferences and Time-Dependent Continuous Costs". In Lee McCluskey, Brian Williams, José Reinaldo Silva, Blai Bonet (Eds.): *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling*, AAAI Press (ICAPS'12), 2–10.

[23] Bezrucav, Stefan-Octavian; Corves, Burkhard (2022) "Modelling Automated Planning Problems for Teams of Mobile Manipulators in a Generic Industrial Scenario". *Appl. Sci.*, **12**, 2319. https://doi.org/10.3390/app12052319

[24] Cashmore, Michael, Fox, Maria, Long, Derek, Magazzeni, Daniele, Ridder, Bram, Carrera, Arnau et al. (2015), "ROSPlan: Planning in the Robot Operating System". In Ronen Brafman (Ed.) *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, Held 7 - 11 June 2015 in Jerusalem, Israel. Palo Alto, Calif.: AAAI Press, 333-–341.

[25] Breiman, Leo (2001) "Random Forests", in *Machine Learning*, Springer, **45**:5-–32, doi: 10.1023/A:1010933404324.

[26] Illingworth, J., Kittler, J. (1987) "The Adaptive Hough Transform", in *Transactions on Pattern Analysis and Machine Intelligence*, IEEE, **5**:690-–698, doi: 10.1109/TPAMI.1987.4767964.