applied sciences

MDPI

Article

A Toolchain for Automated Control and Simulation of Robot Teams in Carbon-Fiber-Reinforced Polymers Production

Marian Körber [1] and Roland Glück [2,*]

[1] Franka Robotics, 80797 München, Germany; marian.koerber@posteo.de
[2] German Aerospace Center, 86159 Augsburg, Germany
* Correspondence: roland.glueck@dlr.de

Abstract: This paper introduces, as a proof of concept, a tool chain for automated control and simulation of a robot team in the domain of production of carbon-fiber-reinforced polymers. The starting point is a CAD construction of a simple aviation component from which single cut pieces of carbon fiber, together withtheir properties, are extracted. Using this information and the layout of a given robot cell, various possibilities of assignments of cut pieces to grippers and robots or robot teams are determined. Subsequently, two approaches using an PDDL solver are introduced, with the goal of finding a scheduling for the lay-up process. Finally, the resulting process is simulated using a physics and rendering engine. The main purpose of this paper is to show the feasibility of such an approach; we do not concentrate on the optimization of single process steps and other details. Due to the modular structure of our approach, extensions and optimizations of the single blocks are easy to integrate. At the moment, digitization and automated control are little explored areas in the domain of production technology using pick and place processes in the aerospace industry. We think that our work will lead to further research in this direction.

Keywords: automation; control; production; robot teams; carbon-fiber-reinforced polymers; planning domain definition language

check for updates

Citation: Körber, M.; Glück, R. A Toolchain for Automated Control and Simulation of Robot Teams in Carbon-Fiber-Reinforced Polymers Production. *Appl. Sci.* **2024**, *14*, 2475. https://doi.org/10.3390/app14062475

Academic Editor: Yutaka Ishibashi

Received: 6 February 2024
Revised: 9 March 2024
Accepted: 11 March 2024
Published: 15 March 2024

1. Introduction

In recent years, pick and place processes in aerospace industry have attracted a lot of interest both in research and industry, which can be seen, e.g., in [1–3]. Another area of research with a long history concerns generating robot programs and control of robots in an automated way based on the layout of a robot cell and CAD data of the tooling. In particular, the work in [4,5] deals with pick and place processes in the production of parts made of carbon-fiber-reinforced polymers (CFRP) in aerospace industry. There, the main approach is always the following one: flat cut pieces (also called plies) of carbon fiber are picked up from a flat table using grippers mounted on robots. Subsequently, these pieces are put into a possibly curved mold. In every case, either the positions where the gripper picks up the flat cut piece (so-called grip points) from the table or drops the cut piece (so-called drop points) into the mold are determined. A system of correspondences between the points on the flat table and the mold, automatically generated by a suitable script from the CAD data, is used to compute the drop positions from the grip positions or vice versa. Using a vision system, cut pieces can be detected on the table and handled in accordance with the aforementioned information.

All of this work focuses on real production and process issues, but is not concerned with a priori simulation and, moreover, shows little flexibility in terms of scheduling the process steps. Here, we provide in addition new ideas and techniques, and try to tackle the following issues:

- Order of the laying steps: In the work cited above, the cut pieces are laid stolidly in the same order as they appear in the CAD file. Here, we try to identify only geometric dependencies influencing the possible order of the laying process, which will result in a partial order of the process steps instead of a rigorous linear order as before. This leads to a greater flexibility of the process planning, creating space for optimization and speed-up of the process.

- Parallel execution: In the work mentioned above, no parallelization of robot actions was considered. Here, we consider also parallel execution of tasks, so one robot can pick up a cut piece from the table while another robot places a cut piece into the mold or moves back from the mold to the table.
- Flexible assignment: While the work cited above assigns one gripper or one gripper group to each cut piece, we allow multiple gripper configurations for one ply. This also increases flexibility of process planning and enables steps towards optimization.

Although all these extensions increase the complexity of planning the process, they increase its flexibility at the same time.

Also, we provide the possibility of simulating the process a priori before execution on a real physical robot cell. Using a mighty physics engine, this can protect again surprises which was never considered in the work mentioned above. In particular, we also model the behavior of the flexible and bendable CFRP material in addition to the rigid bodies of robots, desks and forms.

We follow an approach closely related to the one described in [6], which also contains a description of the use case. The approach consists basically of four stages: the first one is analysis and decomposition of the construction plan, the second one deals with assignment of process steps to tools, the third one is dedicated to scheduling and planning, and the final stage concerns simulation or execution. The analysis and assignment phases correspond in the present work to Section 3, the scheduling and planning phase to Section 4, and simulation or execution (in our case, we only consider simulation) is represented by Section 5.

None of the single steps is of great novelty; however, in the context of CFRP production technology, there is no similar work. Even nowadays, pick and place processes of CFRP production in aerospace industry are carried out manually, and show a low level of automation. This can be underlined by the fact that a search in Scopus using the search parameters "cfrp AND pick AND place" only yields six results. Moreover, none of this work is concerned with scheduling or simulation which shows the motivation behind our work, aiming to develop an application-oriented framework. So the tool chain established here is expected to give raise to new ideas for automation and digitization in this area. In particular, traditional offline programming tools for robots do not take the behavior of textile materials into account. Of course, one can not expect that the simulation of a physics engine will model the real behavior correctly; however, it can give valuable hints whether a process is executable or not in its intended form.

A lot of work deals with single aspects of our approach. The aspect of offline robot programming, based on CAD data, is covered, e.g., in [7,8]. The problem of assigning resources to tasks is the subject of [9,10] whereas [11,12] deals with scheduling problems involving robots. A generic approach for planning assembly sequences is given in [13]; however, this work does not take into account the simulation of material properties.

The present work is an extended version of [14]; in particular, it is accompanied additionally by the final result of the simulation (see also the Supplementary Materials). The rest of the paper is organized as follows: Section 2 gives a short introduction to design and production of CFRP parts in our context, and describes the layout of the robot cell we use for our simulation. In Section 3, we describe how we extract information from the construction plan and how this information is used for the assignment of tools (in our case, grippers mounted on a robot) to process steps (here picking and placing of carbon fiber matrices). Section 4 shows how the assignments from the previous section can be used to formalize the process steps by means of a description in the so-called Planning Domain Definition Language (PDDL). The simulation of the process obtained from a solution of the scheduling problem from the previous section is described in Section 5, followed by a short overview of the overall tool chain in Section 6. Finally, the results are given in Section 7, and a conclusion and directions of future work are sketched in Section 8.

2. Set Up

In this section, we describe one way to design and manufacture CFRP parts in the aviation industry. In Section 2.1, we describe the construction and production process of CFRP parts, whereas Section 2.2 deals with the layout of the robot cell.

2.1. CFRP Construction and Production

One basic way to produce CFRP parts is draping carbon fiber cut pieces into a mold, followed by the infusion of a suitable resin and subsequent hardening in an oven or autoclave. A good overview over such processes is given, e.g., in [15]. As a use case, we chose as a mock-up a cylinder segment shaped tooling similar to a part of an aircraft fuselage. Into this mold, various carbon fiber cut pieces have to be laid from a flat table. A so-called plybook indicating the geometries of the plies was created in CATIA, a standard tool for construction of CFRP components. The plies are constructed in their goal position in the form (which are in general three-dimensional geometries); a CATIA routine called "flattening" unwinds them to flat pieces of two-dimensional geometry (which will be picked up from a flat table). All together, the sample lay-up used throughout this paper consists of 23 carbon plies of different sizes and forms, which are partially overlapping in their final positions in the form. A screenshot from the construction is given in Figure 1.

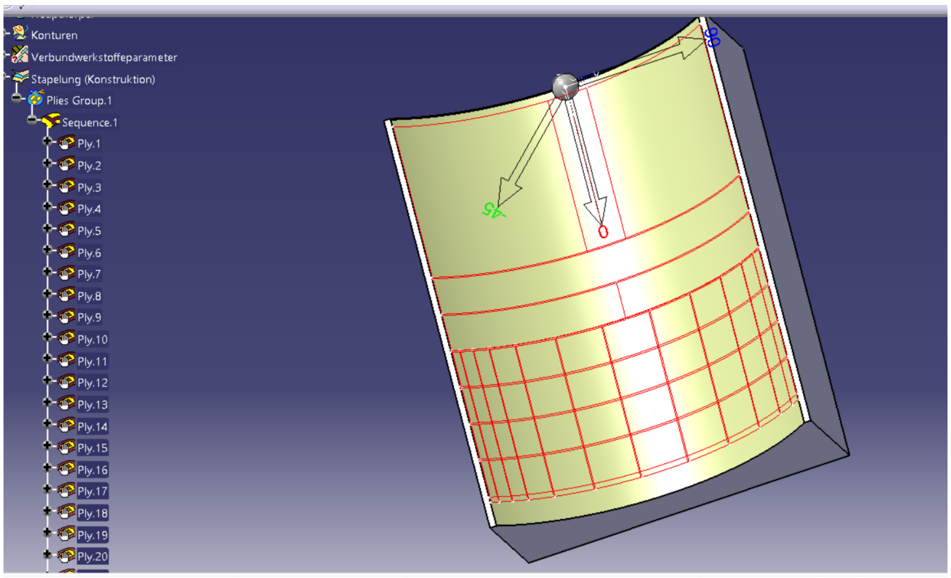


Figure 1. Example ply book

A schematic cross section through a resulting lay-up is shown in Figure 2 where a total of five cut pieces is placed in two layers.

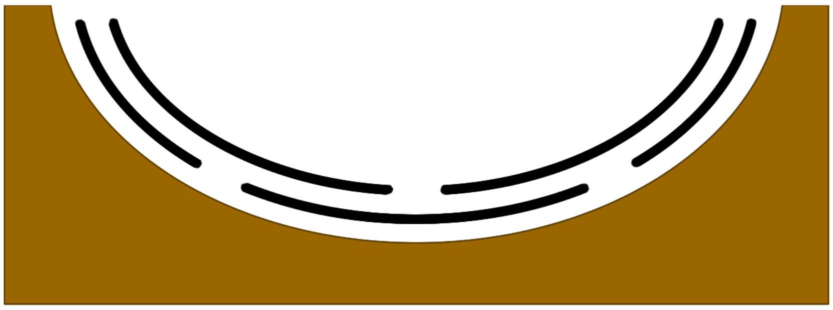


Figure 2. Cross section of a lay-up.

2.2. Robot Cell Layout

As mentioned earlier, the cut pieces from the plybook are placed into the form using grippers mounted on robots. The design of the production cell was inspired by similar plants at the facilities of the German Aerospace Center in Augsburg, which are described, e.g., in [16,17]. There, two upright KUKA robots are mounted on two parallel linear axes, allowing them to move from the table (where the flat cut pieces are supplied) to the mold and back. Both of the robots are equipped with identical grippers (called "snake grippers"), which are real existing grippers at the German Aerospace Center in Augsburg, already used in practice as described in [4]; similar configurations of this kind were investigated also, e.g., in [18,19]. These grippers have vacuum suction devices on their bottom, which can grasp carbon fiber textiles and even metal sheets; compare also [20] for an approach how to handle carbon fiber material using such kind of gripping devices. An additional property of these grippers is that they are also deformable along their longitudinal axis (hence their naming as "snake grippers"); however, this ability is of no importance here, since we place the cut pieces also along the longitudinal axis of the cylindrical mold.

Figure 3 shows this scenario with the robots at the right and the left with the grippers mounted on them, carrying already a cut piece in cooperation. The desk from where the plies are picked up can be seen partially in the foreground.

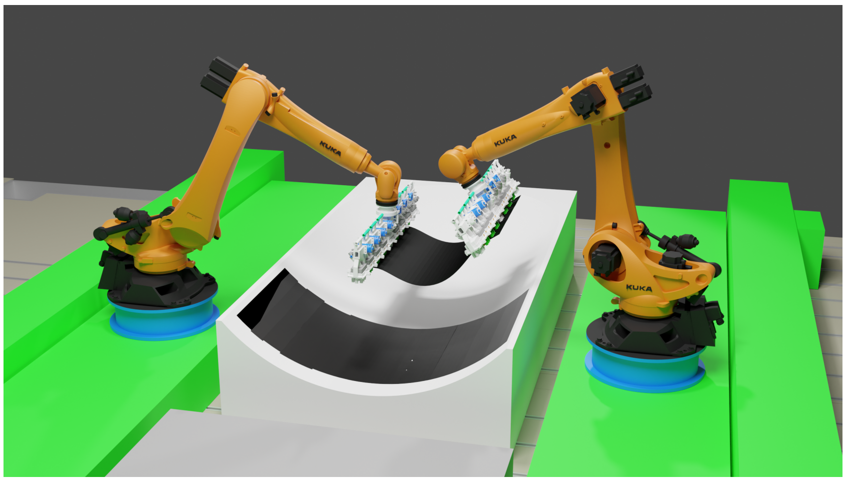


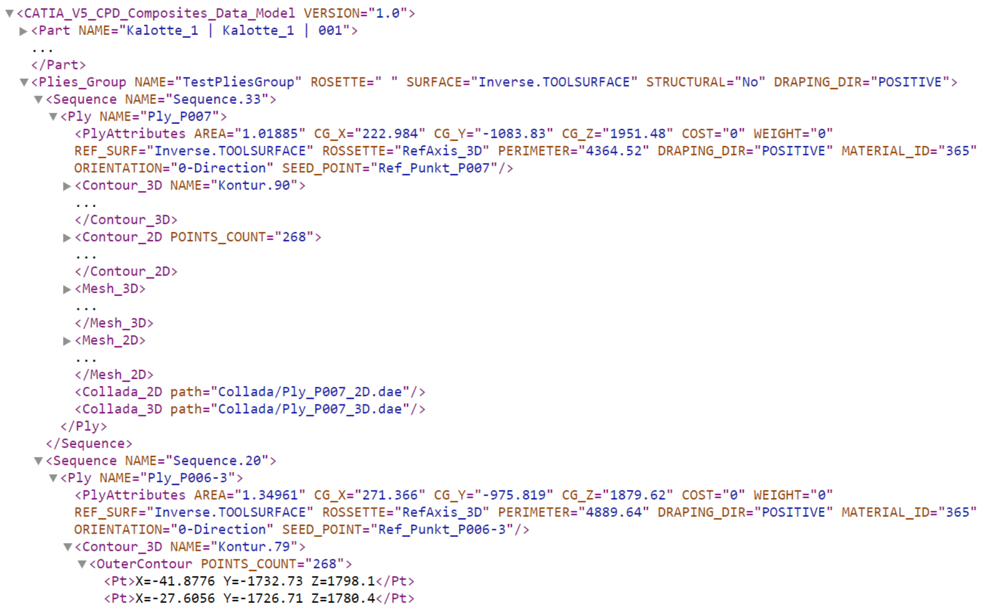
Figure 3. The cell layout.

3. Modeling and Precomputation

Before we can use automatic planning tools, we have to make some considerations how to model the process, and how to extract information from the plybook and the cell layout. All the steps described in this section are based on an XML export from CATIA, which contains all information needed, in particular the shape of the plies on the picking table and the final position of the plies in the form. Moreover, this XML export contains also the material the plies are made of.

An impression of this XML export is given in Figure 4. The `Part` entry (which is not opened here for reasons of better readability) contains, among other things, information about the material and coordinate systems. A `PliesGroup` entry consists of several `Sequence` entries with the associated `Ply` entries which, in turn, contain information about the ply's geometry (in `Contour2D` and `Contour3D`) and correspondences between points on the table and in the form (by means of `Mesh2D` and `Mesh3D`).

One basic step in our modeling is the assignment of grippers to plies, i.e., to determine which gripper or which gripper teams (in our case, each robot is equipped with one gripper so a team of two grippers involves also two robots; see also Figure 3) can be used to pick and place a ply under consideration. In order to assign grippers or gripper teams to plies, we have to determine some geometric characteristics of a ply-like shape, curvature, and size, as well as the ply's material. Conversely, grippers are classified by geometric attributes like width, height, and deformability, indicating whether they can be used for picking and placing plies of given shape and goal curvature. Other attributes of a gripper are its picking method, i.e., whether it uses vacuum gripping, volume flow, or needles, its weight (which is important for the distribution of grippers to robots) and its load capacity (which is rarely of importance in the field of CFRP production). The gripping method is associated with the material; so, an air permeable ply cannot be handled by a vacuum gripper. In our setup, the present grippers can deal with the material from the plybook; however, our approach is expected to also deal with plybooks with varying materials, since the derived actions abstract from these conditions.



```
▼<CATIA_V5_CPD_Composites_Data_Model VERSION="1.0">
 ▶<Part NAME="Kalotte_1 | Kalotte_1 | 001">
  ...
  </Part>
 ▼<Plies_Group NAME="TestPliesGroup" ROSETTE=" " SURFACE="Inverse.TOOLSURFACE" STRUCTURAL="No" DRAPING_DIR="POSITIVE">
   ▼<Sequence NAME="Sequence.33">
     ▼<Ply NAME="Ply_P007">
        <PlyAttributes AREA="1.01885" CG_X="222.984" CG_Y="-1083.83" CG_Z="1951.48" COST="0" WEIGHT="0"
        REF_SURF="Inverse.TOOLSURFACE" ROSSETTE="RefAxis_3D" PERIMETER="4364.52" DRAPING_DIR="POSITIVE" MATERIAL_ID="365"
        ORIENTATION="0-Direction" SEED_POINT="Ref_Punkt_P007"/>
       ▶<Contour_3D NAME="Kontur.90">
        ...
        </Contour_3D>
       ▶<Contour_2D POINTS_COUNT="268">
        ...
        </Contour_2D>
       ▶<Mesh_3D>
        ...
        </Mesh_3D>
       ▶<Mesh_2D>
        ...
        </Mesh_2D>
        <Collada_2D path="Collada/Ply_P007_2D.dae"/>
        <Collada_3D path="Collada/Ply_P007_3D.dae"/>
      </Ply>
    </Sequence>
   ▼<Sequence NAME="Sequence.20">
     ▼<Ply NAME="Ply_P006-3">
        <PlyAttributes AREA="1.34961" CG_X="271.366" CG_Y="-975.819" CG_Z="1879.62" COST="0" WEIGHT="0"
        REF_SURF="Inverse.TOOLSURFACE" ROSSETTE="RefAxis_3D" PERIMETER="4889.64" DRAPING_DIR="POSITIVE" MATERIAL_ID="365"
        ORIENTATION="0-Direction" SEED_POINT="Ref_Punkt_P006-3"/>
       ▼<Contour_3D NAME="Kontur.79">
         ▼<OuterContour POINTS_COUNT="268">
            <Pt>X=-41.8776 Y=-1732.73 Z=1798.1</Pt>
            <Pt>X=-27.6056 Y=-1726.71 Z=1780.4</Pt>
```

Figure 4. The XML export from CATIA.

Rectangular plies with the longer side parallel to the cylindrical longitudinal axis below a given threshold size, depending on the gripper's width, can be handled by one single gripper; in our scenario, every gripper can handle such a ply. In order to position the gripper on the ply, we compute the smallest enclosing rectangle and position the gripper parallel to this rectangle's sides in the middle of the ply. For greater simply curved plies, we use two such grippers, positioned at the rims of the sides of the smallest enclosing rectangle, parallel to the longitudinal axis of the form. If the goal position of the ply is doubly curved, we have to use other grippers which can be positioned using already existing scripts; see, e.g., [21] for this topic. However, in the use case described here, there are no doubly curved plies due to the nature of the chosen form.

Another important point concerns the order in which the plies have to be placed in the form. Traditionally, the plies are placed in the form in the same top-down linear order in which they appear in the plybook, as performed, e.g., in [4,5]. However, this order may be random, and not justified by geometric or process reasons. For example, two symmetrical non-overlapping plies can be placed in any order which, in each case, results in a permissible process. We express this by the computation of a so-called dependency matrix, a two-dimensional Boolean matrix in which the entry at position `[i][j]` indicates whether the `i`-th ply has to be placed before the `j`-th ply. The entry at position `[i][j]` is true if the following two conditions hold:

- The `i`-th and th `j`-th ply overlap in the form, and
- the `j`-th ply lies above the `i`-th ply in the form.

Note that this construction may overlook transitivity, i.e., it could be the case that the first ply has to be placed before the second one (because they overlap and the first ply lies under the second one) by the given criteria, and that the second one has to be placed before the third one (by an analogous argument). However, it could be the case that the first and the third ply do not overlap. This can be overcome by simply computing the transitive hull of the obtained Boolean dependency matrix. This computation may not be necessary in order to use the tools described in the next section (PDDL solvers will derive such transitive dependencies on their own); however, we use this matrix including transitive dependencies to determine which plies can be placed immediately after each other.

To this end, we simply compute the Hasse diagram of this transitive matrix (recall that the Hasse diagram of an order relates direct successors of this order to each other, see, e.g., [22,23] for details). In this matrix, the entry `[i][j]` is true if the `i`-th ply can be placed immediately before the `j`-th ply. The purpose of this information is to enable parallel execution of picking and dropping. For example, consider the case that both the `i`-th and the `j`-th ply can be picked and dropped by every robot, and that the `j`-th ply can be placed immediately after the `i`-th ply. Then, we can pick the `j`-th ply with one robot at the same time that we place the `i`-th ply with another robot. In general, this will lead to a faster overall process than a process consisting only of pick and drop process steps.

4. Approaches

In this section, we introduce and compare different approaches for the scheduling of actions in the described scenario. We stress again that this is not the main purpose of this work, but only a means to an end to demonstrate the whole tool chain.

4.1. Traditional Approach

As already mentioned a few times before, the most common approach in the scenario under consideration is to process the plybook ply by ply in exactly the same order in which the plies appear in the plybook. A manual placement of grippers on plies for the picking step is accompanied by an automated computation of the associated drop points (i.e., the points where the grippers have to be positioned in order to place the ply on the desired position in the form) and offline trajectory planning. Examples for this approach are described in [4,5], or in the context of fiber metal laminate in [24]. Here, the overall process is first partitioned into a linear sequence of steps associated with each ply which, in turn, are broken down to the pick, the transport, and the place process. Of course, the mentioned work does not aim for optimal scheduling, but rather deals with offline programming and process issues in real world applications. Moreover, the behavior of the material is not taken into account.

4.2. Using PDDL

In the following, we describe two approaches using the Planning Domain Definition Language (PDDL, see, e.g., [25]). It was introduced 1998 by McDermott for the International Planning Competition (IPC, see [26]), which developed into a regular event, setting benchmarks in automated planning.

PDDL models planning problems in two files: one domain file and one problem file. The domain file describes the data types occurring in the setting, introduces predicates and (in advanced versions of PDDL) numerical functions, and defines actions based on the involved resources and the actions' preconditions and effects. In the problem file, a concrete instantiation of the domain file is given, consisting of instances of the objects, an initial state, and a goal state. Both the initial and the goal state are described in terms of the predicates defined in the domain file. Also, advanced versions of PDDL allow a metric which can be used to formalize the minimization of the overall duration of the process as a goal for the solution.

There is an enormous list of planners for solving problems defined in PDDL, see, e.g., [27] for an alphabetically ordered list. To find a planner suitable for our problem, we had to look for a solver capable of dealing with metrics as described above. A solver fulfilling all these criteria is the ENHSP solver from [28], which we used for our planning. However, every other solver with the ability to deal with metrics can be used for this part.

4.2.1. Simple Approach

The first and rather raw approach is basically a PDDL version of the ideas described in Section 4.1. In this case, we have as actions only the picking and placing of a ply and the drive of the robot to the table (the drive from the table to the form is included in the placing part). Depending on the robots associated with each ply, as described in Section 3, we have the action that one robot picks and drops a ply, or that both robots have to cooperate in order to fulfill these tasks. All together, we have the following actions:

- Picking of a ply by one robot;
- Picking of a ply by two robots;
- Placing of a ply by one robot;
- Placing of a ply by two robots;
- Drive at the table of one robot;
- Drive at the table of two robots.

Of course, all of these actions have associated preconditions and effects in our PDDL encoding. For example, a ply can be picked only if the robot stands at the table and does not hold another ply. Another example is that a ply can be placed only if the robot holds the ply and all other plies that have to be placed before are already placed (this information stems from the dependency matrix described in Section 3). Conversely, the effect of picking a ply is that the robot holds a ply, and the effect of placing a ply is both that the robot does not hold a ply any more and that the ply is placed in the form. Additionally, each action was assigned a duration increasing the overall duration after its execution. For simplicity, we abstracted here from concrete times and chose one time unit for the picking and driving actions, and two time units for the placing (which also includes the drive from the table to the form). This corresponds roughly to real time ratios; since we are concerned with simulation only as final goal, this is completely sufficient for our purposes. As an example, Figure 5 gives the PDDL representation for the picking of a ply (here Ply5) with two robots.

```
(:action pick-Ply5-rob12
    :parameters (?p - Ply5 ?q - rob1  ?r - rob2)
    :precondition (and (not(busy1 ?q)) (not(busy2 ?r)) (at-table1 ?q)
       (at-table2 ?r))
    :effect (and (Ply5-pickedby-rob2 ?p ?r) (Ply5-pickedby-rob1 ?p ?q)
       (busy2 ?r) (busy1 ?q) (increase (overallduration) 1.0))
)
```

Figure 5. PDDL action for picking.

We abstract from the condition that a ply is at the table because we focus on the pick and place process. Moreover, we assume that a ply can be delivered at the table at random by a suitable conveyor system without affecting the rest of the setup.

In the problem file, we defined the initial state where both robots stand at the table, no ply is placed in the form, and the overall duration equals zero. In the goal state, every ply has to be placed at its position in the form, and the robots have to be back at the table. Of course, we demanded in the metric part of the problem file the minimization of the overall duration.

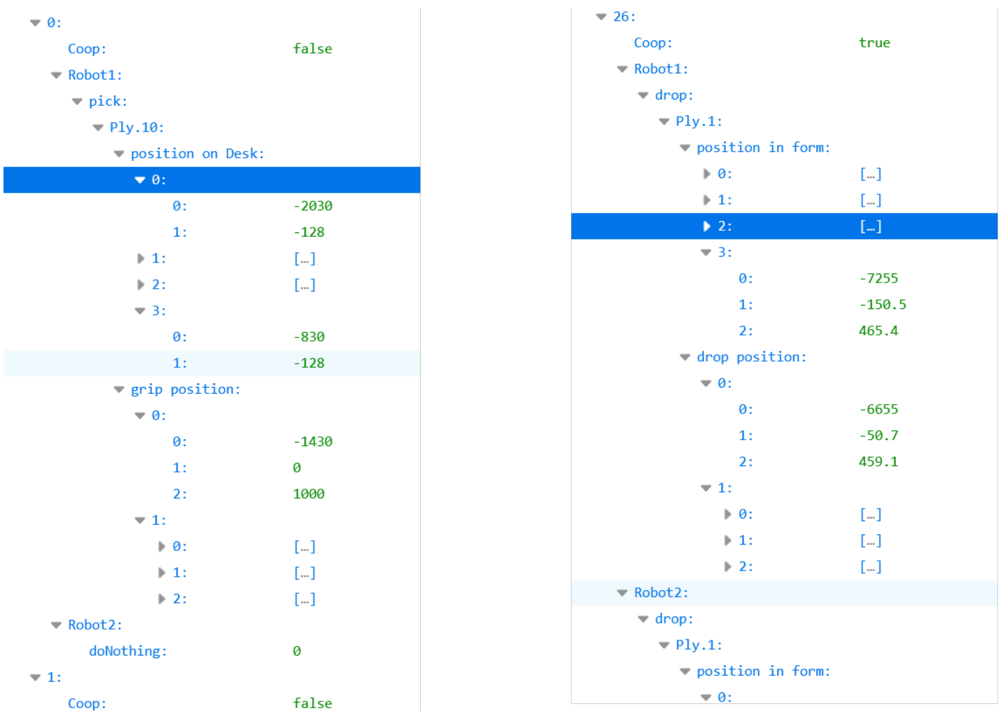
4.2.2. Using Parallelization

This approach extends the previous one and takes into account the possibility of parallel execution of tasks. The problem file is the same as described above, but the domain file contains additional actions, since we consider also simultaneous picking and placing. In particular, we additionally introduce the following actions modeling parallel executions:

- Parallel picking of one ply by one robot and placing of another ply by the other robot;
- Parallel placing of a ply by one robot and drive at the table of the other robot;
- Parallel drive of both robots at the table.

Note that there is no parallel picking or placing of two plies. The first two actions involving placing of a ply have an duration of two time units, the third action has a duration of one time unit only. Parallel picking is impossible since we stipulate that there is always at most one ply on the table. The modeling of parallel placing is not necessary because as a precondition for parallel placing, we would have two robots, each of them holding already a ply. Since parallel picking was ruled out, one robot has to remain at the table during the picking process of the other robot. However, in our modeling, the robot already holding a ply could perform a placing action parallel to the picking action of the other robot.

Figure 6 shows cutouts of the JSON of the plan, already enriched with geometric information about gripping and dropping, which will be used for the simulation described in the following Section 5. The left part shows an action performed by one robot alone which picks up a ply from the table. The position on the desk is determined by four two-dimensional coordinates, indicating the vertices of the ply, which will be used for the simulation of the ply. Additionally, the grip position determines the tool center point of the gripper. Here, the first component gives the position of the tool center point in three-dimensional coordinates, whereas the second component corresponds to its orientation as a three-dimensional rotation matrix. Similarly, the right part shows the cooperating dropping of a ply into the form. Of course, the coordinates of the vertices of the ply are given here in three-dimensional coordinates, and the second robot has corresponding data about its positions (which we left here for better visibility).



```
▼ 0:
    Coop:                       false
  ▼ Robot1:
    ▼ pick:
      ▼ Ply.10:
        ▼ position on Desk:
          ▼ 0:
               0:               -2030
               1:               -128
          ▶ 1:                  […]
          ▶ 2:                  […]
          ▼ 3:
               0:               -830
               1:               -128
        ▼ grip position:
          ▼ 0:
               0:               -1430
               1:               0
               2:               1000
          ▼ 1:
             ▶ 0:               […]
             ▶ 1:               […]
             ▶ 2:               […]
  ▼ Robot2:
      doNothing:                0
▼ 1:
    Coop:                       false
```

```
▼ 26:
    Coop:                       true
  ▼ Robot1:
    ▼ drop:
      ▼ Ply.1:
        ▼ position in form:
          ▶ 0:                  […]
          ▶ 1:                  […]
          ▶ 2:                  […]
          ▼ 3:
               0:               -7255
               1:               -150.5
               2:               465.4
        ▼ drop position:
          ▼ 0:
               0:               -6655
               1:               -50.7
               2:               459.1
          ▼ 1:
             ▶ 0:               […]
             ▶ 1:               […]
             ▶ 2:               […]
  ▼ Robot2:
    ▼ drop:
      ▼ Ply.1:
        ▼ position in form:
          ▼ 0:
```

Figure 6. Two actions in JSON.

5. Simulation Environment

In our quest to accurately visualize and evaluate the handling of dry carbon fiber tissues by dual robots, we drew upon the pioneering work of Seita et al., who explored novel approaches to manipulating fabric using both model-free and model-based deep learning in simulation environments. Their research, as detailed, e.g., in [29,30], highlighted the significance of advanced textile physics simulation, alongside intuitive robot motion control and high-quality visualization capabilities. This body of work, together with the contributions from Wang and Urbanic (2021) in model-based design and simulation of soft robotic grippers for fabric material handling [31], and Makris et al. (2022) on model-based motion planning for human–robot co-manipulation of deformable objects [32], guided our selection of Blender as the optimal simulation environment. Blender's GPL licensing and its comprehensive physic engine (Bullet), as we will demonstrate, offer a detailed process representation that is crucial for our objectives, as illustrated in Figure 3 where one can see the sag of the carbon fiber cut piece.

The fiber tissue model was calculated using the Particle Method, where each mass point's connections simulate the material's physical properties. To accurately replicate the shear effect of the dry carbon fiber tissues, the connections between each mass point were configured to be as stiff as possible in terms of linear displacement, ensuring the material's tensile strength is realistically modeled. Conversely, the rotational stiffness of these connections was set to be soft, allowing the simulation to accurately capture the material's inherent flexibility, as well as its response to shear forces. This approach enables a nuanced simulation of the carbon fiber's behavior, reflecting both its strength and flexibility under various handling conditions.

The simulation, enabling flexible camera positioning and leveraging Blender's rendering power for optimal visual analysis, outputs to an .avi file which, in turn, can be converted to other formats, see also the Supplementary Materials. Building upon the strategies delineated in Section 4, we exported the planned robotic actions and spatial configurations as JSON files, encompassing each robot's maneuvers and the precise locations for gripping and releasing (during the placing of a ply in the form) the textile plies. We developed a suite of Python scripts to interpret these plans, orchestrating the robot movements and simulating the gripping forces effectively. While the gripping contours were crucial for initializing the handling process, the subsequent transportation and release phases were governed by Blender's physics engine, ensuring a realistic emulation of the textile behavior, as further discussed in Section 7.

Additionally, it is noteworthy that the simulation environment in Blender allows for automatic optimization and validation of the handling processes. However, due to the computationally intensive nature of calculating material behavior, employing Blender for Reinforcement Learning or other trial-and-error optimization methods is not recommended. The significant computational resources required for such approaches make them impractical for this context, underscoring the need for more efficient optimization strategies when working with complex material simulations.

6. Overall Tool Chain

The overall tool chain for our approach, putting all steps described so far together, is depicted in Figure 7: from the CAD file, we extract first the plies and their properties like shape, size, curvature and material. Also, we compute from this information the dependency matrix as described in Section 3. Together with the list of gripper properties, we determine which gripper or which gripper teams can handle which ply. Together with the cell layout, which contains information about the available gripper and gripper teams, we construct a PDDL formulation of our problem, which is subsequently solved. This solution is transformed back (using geometric information from the CAD file and the cell layout) to concrete robot motions, which are fed into Blender for simulation.

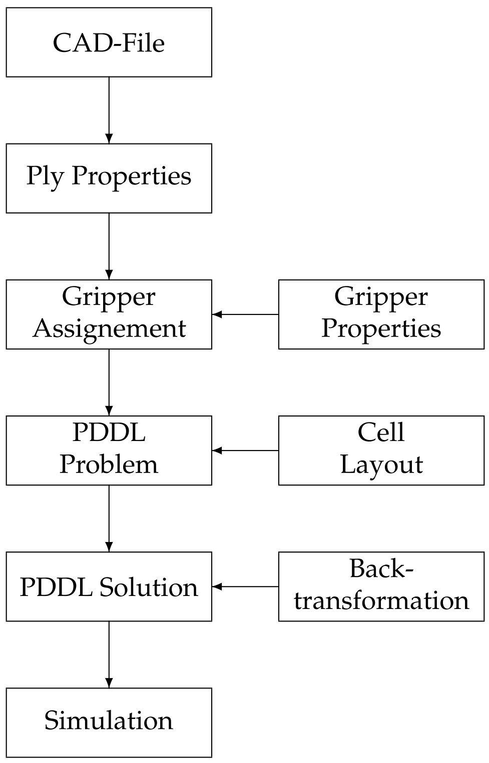


Figure 7. Workflow of the PDDL approach.

7. Results

The main criterion for the evaluation of our approach is the time needed to obtain the final solution (not to be confused with the process time). This time is determined mainly by the time for solving the PDDL instance and the time for the simulation (see below). As an additional observation, we compared the time units needed to complete the construction in our simulation (which we can compare to the traditional approach described in Section 4.1).

The most time-consuming steps in our approach were the solution of the PDDL problem and the rendering in Blender, each taking roughly 30 min on an Intel(R) Core(TM) i7-10850H CPU @ 2.70 GHz using 16 GB of RAM. The other steps could be executed almost instantaneously, and had no influence on the computation time.

As one would expect, the approach described in Section 4.2.2 outperformed the other approaches concerning overall process time: the traditional approach from Section 4.1 and the simple PDDL approach from Section 4.2.1 both needed 103 time units, whereas the parallelized PDDL approach from Section 4.2.2 needed only 82 time units. If we want to transfer these results to physical reality, this has to be seen with the caveat of the immediate furnishing of plies as already mentioned in Section 4.2.1. Moreover, an automated approach saves a lot of working hours compared to the traditional approach, where the scheduling is performed by hand (as used, e.g., in [3,5]).

Other, also insightful conclusions could be drawn from the simulation (and can be useful in future simulations):

- Collisions between the involved robots, desks and forms can be detected and dealt with.
- Due to the physics engine, the behavior of the material can be predicted to a certain extent. This allows to identify situations where the ply may sag too much if the grippers are too close to each other. Conversely, situations where the grippers are directed too far apart lead to a breakaway of the material which also can be seen in the simulation.

8. Conclusions and Outlook

This work showed the first steps towards automation in an area where in real world production manual work is still rampant. Also, planning and scheduling of automated tasks in this domain are still performed in large parts by hand, which is time-expensive and may often lead to sub-optimal solutions. Compared to this old-fashioned approach, our work shows how automation and simulation can be carried out in this domain. Moreover, our approach leads to both shorter process times and savings in human working time needed for the planning and scheduling of the process. Another point is that the simulation of the fiber material can also help to avoid superfluous tests and experiments. We plan to evaluate the presented approach (which was presented here at a relatively early stage) on a real world test case at the DLR facilities in Augsburg. This may also help to find mistakes in the planning and scheduling modules.

Due to the modular character of the approach, as shown in Figure 7, there is plenty of room for experimentation and extensions in various places. We mention the following directions of future work:

- Gripper assignment: Here, we demonstrated the assignment only for a special class of grippers and geometries. However, the approach described in Section 3 can be transferred to grippers of a comparable geometry, i.e., single curved and longish grippers. This holds regardless of the gripping method. For handling doubly curved geometries, both with respect to grippers and goal geometries of plies, approaches as described in [21] can be used. In this work, the grippers are also deformable, and not rigid as in the present paper. The integration into the framework presented here poses no problems; in this case, the greatest challenge here will be the modeling in Blender or another engine.
- Scheduling: In the approach presented here, the scheduling of actions was determined via PDDL (and an associated solver). PDDL solvers are not guaranteed to find an optimal solution in acceptable time due to state explosion so they exploit (powerful) heuristics instead. Other methods as presented in [13,33] could be an interesting field for further investigations. A more recent idea is the employment of quantum computing, as was investigated for similar problems, e.g., in [34].
- Parallelization: In the present work, parallel actions were introduced by hand as described in Section 4.2.2. PDDL 2.1 allows also for parallel actions as described in [35]. Using this feature, the modeling in PDDL could be simplified significantly. Also, these ideas can be integrated in the overall workflow without effort.

Of course, the use of other PDDL solvers, rendering environments and physics engines, should also be investigated. For PDDL solvers, we refer again to the list from [27]; for simulation, we mention WeBots (see [36]), PyBullet (see [37]), Gazebo (see [38]) and MuJoCo (see [39]). To deal with the control of the high degree of freedom of the robots, one can consider methods as developed, e.g., in [40]. Lastly, the validation of the approach on a real world application should be executed to judge its merits.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app14062475/s1, TeamBots Simulation.

Author Contributions: The first author, M.K., carried the main work load of the paper; the second author, R.G., offered support in the area of PDDL. All authors have read and agreed to the published version of the manuscript.

Funding: This work gratefully acknowledges support by the DFG project "TeamBotS".

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to privacy.

Conflicts of Interest: Author Marian Körber was employed by the company Franka Robotics. The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CAD	Computer-Aided Design
CFRP	Carbon-Fiber-Reinforced Polymers
PDDL	Planning Domain Definition Language

References

1. Björnsson, A.; Jonsson, M.; Johansen, K. Automated material handling in composite manufacturing using pick-and-place systems a review. *Robot. Comput.-Integr. Manuf.* **2018**, *51*, 222–229. [CrossRef]
2. Angerer, A.; Ehinger, C.; Hoffmann, A.; Reif, W.; Reinhart, G. Design of an automation system for preforming processes in aerospace industries. In Proceedings of the IEEE Conference on Automation Science and Engineering, CASE 2011, Trieste, Italy, 24–27 August 2011; pp. 557–562. [CrossRef]
3. Eckardt, M.; Buchheim, A.; Gerngross, T. Investigation of an automated dry fiber preforming process for an aircraft fuselage demonstrator using collaborating robots. *CEAS Aeronaut. J.* **2016**, *7*, 429–440. [CrossRef]
4. Schuster, A.; Frommel, C.; Deden, D.; Brandt, L.; Eckardt, M.; Glück, R.; Larsen, L. Simulation Based Draping of Dry Carbon Fibre Textiles with Cooperating Robots. *Procedia Manuf.* **2019**, *38*, 505–512. [CrossRef]
5. Schuster, A.; Larsen, L.; Fischer, F.; Glück, R.; Schneyer, S.; Kühnel, M.; Kupke, M. Smart Manufacturing of Thermoplastic CFRP Skins. *Procedia Manuf.* **2018**, *17*, 935–943. [CrossRef]
6. Glück, R.; Hoffmann, A.; Nägele, L.; Schierl, A.; Reif, W.; Voggenreiter, H. Towards a Tool-based Methodology for Developing Software for Dynamic Robot Teams. In Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2018—Volume 2, Porto, Portugal, 29–31 July 2018; Madani, K., Gusikhin, O., Eds.; SciTePress: Setúbal, Portugal, 2018; pp. 615–622. [CrossRef]
7. Nagele, L.; Macho, M.; Angerer, A.; Hoffmann, A.; Vistein, M.; Schonheits, M.; Reif, W. A backward-oriented approach for offline programming of complex manufacturing tasks. In Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications (ICARA), Queenstown, New Zealand, 17–19 February 2015; pp. 124–130. [CrossRef]
8. Neto, P.; Pires, J.N.; Moreira, A.P. CAD-based off-line robot programming. In Proceedings of the 2010 IEEE Conference on Robotics, Automation and Mechatronics, Singapore, 28–30 June 2010; pp. 516–521. [CrossRef]
9. Pfrommer, J.; Schleipen, M.; Beyerer, J. PPRS: Production skills and their relation to product, process, and resource. In Proceedings of the 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), Cagliari, Italy, 10–13 September 2013. [CrossRef]
10. Björkelund, A.; Malec, J.; Nilsson, K.; Nugues, P. Knowledge and skill representations for robotized production. *IFAC Proc. Vol.* **2011**, *44*, 8999–9004. [CrossRef]
11. Schoen, T.R.; Rus, D. Decentralized robotic assembly with physical ordering and timing constraints. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 5764–5771. [CrossRef]
12. Kim, H.J.; Lee, J.H. Deep Reinforcement Learning with a Look-Ahead Search for Robotic Cell Scheduling. *IEEE Trans. Syst. Man Cybern. Syst.* **2024**, *54*, 622–633. [CrossRef]
13. Thomas, U.; Wahl, F. A system for automatic planning, evaluation and execution of assembly sequences for industrial robots. In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180), Maui, HI, USA, 29 October–3 November 2001; Volume 3, pp. 1458–1464.
14. Glück, R.; Körber, M. Automated Control and Simulation of Dynamic Robot Teams in the Domain of CFK Production. *arXiv* **2022**, arXiv:2210.11213.
15. Campbell, F. *Manufacturing Processes for Advanced Composites*; Elsevier: Amsterdam, The Netherlands, 2003; pp. 1–517. [CrossRef]
16. Krebs, F.; Larsen, L.; Braun, G.; Dudenhausen, W. Design of a multifunctional cell for aerospace CFRP production. *Lect. Notes Mech. Eng.* **2013**, *7*, 515–524. [CrossRef]
17. Krebs, F.; Larsen, L.; Braun, G.; Dudenhausen, W. Design of a multifunctional cell for aerospace CFRP production. *Int. J. Adv. Manuf. Technol.* **2016**, *85*, 17–24. [CrossRef]
18. Alebooyeh, M.; Wang, B.; Urbanic, R.J.; Djuric, A.; Kalami, H. *Investigating Collaborative Robot Gripper Configurations for Simple Fabric Pick and Place Tasks*; SAE Technical Papers; SAE International: Warrendale, PA, USA, 2019. [CrossRef]
19. Djuric, A.M.; Rickli, J.; Urbanic, R. A Framework for Collaborative Robot (CoBot) Integration in Advanced Manufacturing Systems. *SAE Int. J. Mater. Manuf.* **2016**, *9*, 457–464.
20. Fleischer, J.; Förster, F.; Crispieri, N.V. Intelligent gripper technology for the handling of carbon fiber material. *Prod. Eng.* **2014**, *8*, 691–700. [CrossRef]
21. Körber, M.; Frommel, C. Automated Planning and Optimization of a Draping Processes Within the CATIA Environment Using a Python Software Tool. *Procedia Manuf.* **2019**, *38*, 808–815. [CrossRef]
22. Grätzer, G. *Lattice Theory: Foundation*; Springer: Basel, Switzerland, 2011.
23. Roman, S. *Lattices and Ordered Sets*, 1st ed.; Springer: New York, NY, USA, 2008.

24. Vistein, M.; Deden, D.; Glück, R.; Schneyer, S. Automated Production of Large Fibre Metal Laminate Aircraft Structure Parts. *Procedia Manuf.* **2019**, *38*, 1300–1307. [CrossRef]
25. Planning Wiki. Available online: https://planning.wiki (accessed on 20 December 2023).
26. International Conference on Automated Planning and Scheduling. Available online: https://www.icaps-conference.org/competitions/ (accessed on 20 December 2023).
27. Planners from A to Z. Available online: https://planning.wiki/ref/planners/atoz (accessed on 20 December 2023).
28. The ENHSP Planning System. Available online: https://sites.google.com/view/enhsp/ (accessed on 20 December 2023).
29. Hoque, R.; Seita, D.; Balakrishna, A.; Ganapathi, A.; Tanwani, A.K.; Jamali, N.; Yamane, K.; Iba, S.; Goldberg, K. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In Proceedings of the Robotics: Science and Systems XVI, Corvalis, OR, USA, 12–16 July 2020. [CrossRef]
30. Seita, D.; Ganapathi, A.; Hoque, R.; Hwang, M.; Cen, E.; Tanwani, A.K.; Balakrishna, A.; Thananjeyan, B.; Ichnowski, J.; Jamali, N.; et al. Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 9651–9658. [CrossRef]
31. Wang, B.; Urbanic, R.J. *Model-Based Design and Simulation of a Soft Robotic Gripper for Fabric Material Handling*; Research Square: Durham, NC, USA, 2021. [CrossRef]
32. Makris, S.; Kampourakis, E.; Andronas, D. On deformable object handling: Model-based motion planning for human-robot co-manipulation. *CIRP Ann.* **2022**, *71*, 29–32. [CrossRef]
33. Hoffmann, A.; Nägele, L.; Reif, W. How to find assembly plans (fast): Hierarchical state space partitioning for efficient multi-robot assembly. In Proceedings of the Fourth IEEE International Conference on Robotic Computing, IRC 2020, Taichung, Taiwan, 9–11 November 2020; pp. 172–177. [CrossRef]
34. Baioletti, M.; Oddi, A.; Rasconi, R. Solving Scheduling Problems with Quantum Computing: A Study on Flexible Open Shop. In Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO 2023, Companion Volume, Lisbon, Portugal, 15–19 July 2023; Silva, S., Paquete, L., Eds.; ACM: New York, NY, USA, 2023; pp. 2175–2178. [CrossRef]
35. Fox, M.; Long, D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.* **2003**, *20*, 61–124. [CrossRef]
36. WeBots Homepage. Available online: https://cyberbotics.com/ (accessed on 20 December 2023).
37. PyBullet Homepage. Available online: https://pybullet.org/wordpress/ (accessed on 20 December 2023).
38. Gazebo Homepage. Available online: https://gazebosim.org/home (accessed on 20 December 2023).
39. MuJoCo Homepage. Available online: https://mujoco.org/(accessed on 20 December 2023).
40. Nuchkrua, T.; Kornmaneesang, W.; Chen, S.L.; Boonto, S. Precision contouring control of 5 DOF dual-arm robot manipulators with holonomic constraints. In Proceedings of the 2017 11th Asian Control Conference (ASCC), Gold Coast, QLD, Australia, 17–20 December 2017; pp. 976–981. [CrossRef]