Robotics and Computer Integrated Manufacturing 67 (2021) 102020

Contents lists available at ScienceDirect

Robotics and Computer Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Automated planning for robotic layup of composite prepreg

Rishi K. Malhan[a], Aniruddha V. Shembekar[a], Ariyan M. Kabir[a], Prahar M. Bhatt[a], Brual Shah[a], Scott Zanio[b], Steven Nutt[c], Satyandra K. Gupta[*,a]

[a] Viterbi School of Engineering, University of Southern California, Los Angeles, CA, USA
[b] Hexagon Manufacturing Intelligence, India
[c] M.C Gill Composites Center, University of Southern California, Los Angeles, CA, USA

ARTICLE INFO

Keywords:
Composite automation
Automated layup
Robot base placement
Grasp planning
Multi-arm manipulation
Path-constrained trajectory planning
Industrial robots

ABSTRACT

Hand layup is a commonly used process for making composite structures from several plies of carbon-fiber prepreg. The process involves multiple human operators manipulating and conforming layers of prepreg to a mold. The manual layup process is ergonomically challenging, tedious, and limits throughput. Moreover, different operators may perform the process differently, and hence introduce inconsistency. We have developed a multi-robot cell to automate the layup process. A human expert provides a sequence to conform to the ply and types of end-effectors to be used as input to the system. The system automatically generates trajectories for the robots that can achieve the specified layup. Using the cell requires the automated generation of different types of plans. This paper addresses two main planning problems: (a) generating plans to grasp and manipulate the ply and (b) generating feasible robot trajectories. We use a hybrid-physics based simulator coupled with a state space search to find grasp plans. The system employs a strategy that applies constraints successively in a non-linear optimization formulation to identify suitable placements of the robots around the mold so that feasible trajectories can be generated. Our system can generate plans in a computationally efficient manner, and it can handle a wide variety of complex parts. We demonstrate the automated layup by conducting physical experiments on an industry-inspired mold using the generated plans.

1. Introduction

Composites are widely used in the industry to realize lightweight structures with high performance. The global composites industry was valued at $66 billion in 2015 and is expected to grow to $130 billion by 2024 [33]. Carbon fiber is a widely used raw material for the production of composites. For low-volume production and complex parts, a hand layup process is used to laminate plies of carbon fiber prepreg (a resin-infused fiber bed). An expert technician deposits pre-cut plies of prepreg in prescribed orientations on a mold to build up the desired thickness. The layup is then vacuum-bagged and cured at a prescribed temperature and pressure to produce the final component. Fig. 1 illustrates the layup process on a medium-sized mold. The hand layup process is ergonomically challenging and skill-intensive. Human operators must apply various levels of pressure to the plies. Multiple operators must collaborate to conform larger plies to complex contours. The hand layup process is labor-intensive and can exhibit inconsistency due to variability in human operation.

Current automation solutions in the composites industry are limited primarily to automated tape layup (ATL) and automated fiber placement (AFP). In these processes, prepreg tape of various widths (comprised of unidirectional fibers) is used, as opposed to plies of fabric prepreg. These solutions are limited primarily to larger and simpler geometries due to the large tape-dispensing end-effectors. We have shown the feasibility of hand layup automation in our prior work in [59,62] which shows a robotic cell to handle and conform to prepreg plies. Fig. 2 shows the robotic cell used for automating the process. In an industrial setting, we envision a robotic cell to perform the complex layup process under human supervision.

Multiple articulated robot arms are required to perform the layup operation in the robotic cell. Significant planning challenges arise as a consequence of using robots in the cell. We cannot expect humans to program the robots in the cell. The tool paths for the draping tools and grippers used for the operations of the cell must be automatically planned. Planners that generate the point-to-point motion and path-constrained motions are required. Moreover, manipulating the plies is a complex physics-based motion planning problem. Vision and haptics-based sensors must be integrated into an online plan refinement to

* Corresponding author.
E-mail address: guptask@usc.edu (S.K. Gupta).

https://doi.org/10.1016/j.rcim.2020.102020
Received 25 October 2019; Received in revised form 5 May 2020; Accepted 6 May 2020

Fig. 1. Illustration of hand layup process being done by multiple operators [69].

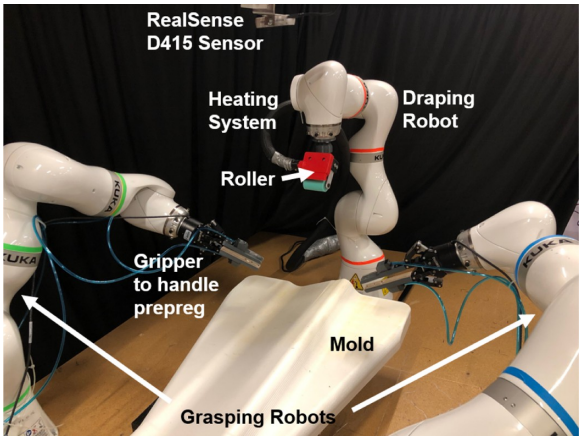

Fig. 2. Robotic cell comprising of manipulator, specialized grippers, heating system, feedback systems, and conforming tools required to automate the carbon fiber ply layup process.

avoid defects and handle contingencies. Path-constrained synchronous trajectories are needed, which satisfy process and dynamic constraints. The feasibility of these trajectories is largely dependent on the placement of the robots around the mold. Planners that can automatically determine where to place the robots around the mold are required. The system must address both trivial and challenging planning problems in order to automate the process.

We identified two planning problems which are challenging for the system: (a) finding grasp plans that generate the tool paths for grippers to handle the ply, and (b) finding suitable robot placements. This work addresses these two planning tasks. Grasp plans are the tool paths for the grippers that manipulate the ply during the process. Several grasp plans can exist in the space around the tool. The system must determine a grasp plan which leads to a defect-free layup and minimizes a cost defined by our approach. Robot placements, on the other hand, influence the feasibility of generating path constraint trajectories. Inappropriate robot placements will lead to failure to execute draping and grasping robot tool paths. We use a state space search strategy assisted by physics-based simulation framework to generate optimal grasp plans. A successive application of constraints strategy is then used to identify feasible robot placements and generate the robot trajectories. We have reported prior work in the robot placement algorithm in [60,61]. In the present study, our work is extended by discussing the challenges that occur when more constraints are imposed on the placement and present a different approach for solving the problem. The two planning problems are computationally challenging, and that our planners can handle complex cases in real-time. We test our algorithms on a mold inspired by an industrial component and also develop interfaces through which humans and robots can collaborate.

2. Related work

2.1. Automation in composite manufacturing

In this section, we describe the progress towards automating the hand layup process.

Previous work [35] demonstrated how a numerical model could be created to predict the draping of a ply on the mold. This drape can then be used to estimate the sequence of motions that can be performed by a technician or a machine to achieve the desired layup.

A categorization of the layup techniques used by technicians has been reported [24]. These layup techniques can serve as baseline motion primitives to automate the ply draping. In [13], a complete system to handle and lay up prepreg on a mold using an industrial robot was proposed. A concept of cell design was illustrated with four manipulation mechanisms. Molfino et al. proposed a hyperflexible cell design in [70]. Although these works lacked algorithms to automatically generate instructions for the robot, they demonstrated concepts of automation solutions that can be applied. Researchers in [25] used an industrial robot to apply pressure to drape the ply. The robot was manually programmed, and the ply was preformed in a hydraulic press to avoid manipulation and shear. This work furnished insights into what end-effectors can be used with manipulators, and challenges in the layup process. Other works also studied the development of end-effectors to handle composites, which requires the design of specialized grippers for carbon fiber prepreg [27,86]. Newell et al. presented a numerical model to predict the behavior of prepreg when gripped at certain points [75]. Behavior or woven prepreg plies on a doubly curved mold was investigated using a kinematic mapping in [47]. Automated handling for pick and place operations of composites during manufacturing was briefly reviewed in [9]. A brief review of grasping technologies and advances in automation of composites is provided in [26].

2.2. Multi-arm manipulation of deformable materials

As illustrated in Fig. 2, the robotic cell requires multi-arm manipulation of a prepreg ply (deformable material). In this section, we will discuss the work related to this topic. Manipulation of flexible materials requires coordinated motion and/or control of multiple different robots [1,18,19,34,37,49,79,87,88,100]. Recent survey papers on the manipulation of deformable objects include [40,44]. Different applications impose different requirements on the underlying approach for solving the problem. Specialized planning, learning, and control approaches have been developed for 1-D problems (e,g., wires and ropes) [50,63,71,77,81–83,92], cloth folding [5,7,22,45,46,54,57,64–67,85,91], and ply manipulation [29,38,48,93]. A method to generate trajectories for multi-robot systems to handle sheet metal while minimizing deformations was proposed in [32]. Some approaches, such as rope manipulation and cloth folding, do not impose strict requirements on the final shapes. These approaches seek only the qualitative presence of the desired feature after the operation. Some applications require the final shape to be controlled precisely. In applications such as composite layup, the behavior of the underlying material can be complex. Therefore, we rely on advanced simulations to estimate the state of the flexible material.

Sampling-based motion planning approaches have been developed for manipulating deformable objects in static environments [6,51,72]. Planning approaches that consider deformable environments have also been investigated [30,31,78,80]. Learning from demonstration methods has been adopted to learn task-specific actions for deformable parts [39,52,53]. Recent work in this area formulates a constrained optimization problem to compute a plan for a team of mobile manipulators to manipulate a highly deformable object [1]. In [21], a team of mobile robots was used to grasp and assemble a part comprised of multiple components. They formulated the problem as a constraint

satisfaction problem, then divided it into independent smaller problems to achieve faster computation times.

2.3. Workpiece/robot placement

This section describes the work related to the robot base placement problem. A variant of this problem is to find locations of components of an assembly [74]. The authors used the gradient of the manipulability index, which is a measure of the ability of the robot to orient the end-effector at a given position. Manipulability index can be obtained from the Jacobian of the manipulator [94–96]. The position and orientation reachability of the robot was precomputed in [90]. The base of the robot was then placed such that target grasping locations could be reached without collision.

These works dealt with individual locations in Cartesian space. A path will consist of a sequence of such locations called waypoints. A path can be represented with respect to a coordinate frame. Finding a feasible placement of this path with respect to the robot is equivalent to finding the placement of the coordinate frame. Aspragathos et al. proposed an approach in [3] in which the mean value of the manipulability index was maximized across all the waypoints in a path. They illustrated this approach by placing a simple curve in the robot workspace. To improve the velocity performance of the manipulator while executing the paths, Aspragathos et al. proposed a manipulator velocity ratio (MVR) [4,73]. MVR can be maximized to obtain the maximum end-effector velocity for the least joint angle velocity, which can improve the velocity performance of the manipulator while executing the path. Cycle time was reduced by placing the workpiece such that it minimized the time coordinated joint motion in [28]. A posture optimization methodology for 6R industrial robots was proposed using performance maps in [55]. Other variants for solving this problem also considered minimizing the energy consumption, actuator torques, and forces [89] for a orthoglide parallel robot.

Authors of [14,15,23] found the placement for a redundant robot. Redundancy in their system arose from the process requiring fewer degrees of freedom (DOF) compared to the DOF of the manipulator. They considered the tool path followed by a milling tool on a small cube. As the orientation of the spindle (end-effector) about its axis of rotation can take any value, it is a redundant system. They formulated the problem with the objective of minimizing the deflection of the spindle during the operation. They presented an approach where the redundant angle was assigned a value while solving the optimization problem. Work in [20] integrated two multi-objective optimization loops. The outer loop was used to fix the redundant angle to minimize joint limits, singularities, and collision as a single non-differentiable objective function. Once redundancy was taken care of, the solution was passed to the inner loop, which determined the workpiece placement. Kabir et al. presented an approach which finds synchronous trajectories for high DOF multi-arm systems in [43]. The pose of robot base frame can be considered in the optimization problem to find suitable base placements.

The trajectory for a high DOF system, including a mobile base, was planned using inverse kinematics branching in [10]. They formulated the problem as a quadratic program, and constraints were included as a part of the objective function. The position of the mobile base could be determined while planning for the trajectory. The overall joint motion was minimized by taking the difference between two consecutive joint configurations on the path in [36]. They formulated an unconstrained optimization problem where redundancy was simultaneously resolved. They also described various local and global indices used in this area as surrogates to serve some objective while placing the path. The work in [97,98] proposes a capability map to encode the position and orientation of the robot in a discrete representation. In [99], they use this map as an approximation to the robot inverse kinematics (IK) to find suitable placements for a mobile base. Hybrid approaches to place the manipulator with respect to the path by avoiding singular configurations have also been studied. For instance, a hybrid method which takes the advantage of genetic algorithms, quasi-Newton optimization, and a constraint handling method was described in [68].

Most of the work revolved around optimizing some local or global performance index for the manipulator. Additional constraints, such as force, velocity, continuity, avoiding singularity, and collision, were either absent or conservatively applied to make the problem computationally tractable. However, this does not ensure that trajectories can actually be generated over the paths. The works cited, including the ones aimed at resolving a redundant system, consider smaller workpieces and simpler geometries. For completely utilizing a robot workspace, an approach is required which can be applied to large complex geometries. Path-constrained trajectory generation is a computationally expensive affair, as it involves identifying the IK of a robot under constraints. If additional constraints like velocity, force, and path continuity are enforced, the problem becomes computationally intractable.

3. Robotic cell for composite layup

3.1. Background

The carbon fiber prepreg ply is a dry fiber bed infused with a viscous thermoset polymer resin (e.g., Fig. 3). Fibers can be unidirectional (UD) or woven fabrics. Depending upon the type of weave and resin system used, various mechanical properties can be obtained. Multiple plies are stacked on top of the mold at prescribed orientations. The designer can tailor the anisotropic mechanical response as per loading conditions by controlling the orientation of each ply and type of materials being used.

The prepreg structure can undergo in-plane deformation (e.g., trellis shear, as shown in Fig. 3). In-plane deformation allows conformation of flat prepreg plies onto non-developable and complex surfaces. Technicians apply local pressure to shear the ply and conform it to the mold. Hand-held tools or hands are used to apply this pressure [42]. Typically, a 1–10 cm^2 area is conformed at a time. Fig. 4 shows the different techniques used during the process and illustrates defects that can occur. Moreover, material and environment-related factors can also influence the drapeability and quality of the layup. Prepreg tack (adhesion), for instance, depends on ambient temperature, humidity, out-time, and through-thickness compliance (bulk factor) of the prepreg. After a layup, the laminate is enclosed in a vacuum bag and cured using a prescribed pressure and temperature cycle to produce the final component.

Quality of the layup is evaluated on the basis of how well the plies are conformed to the substrate (mold or the underlying prepreg). In-plane and out-of-plane shearing can cause changes in fiber orientations. The fiber orientations should be within specified tolerances. In addition, defects such as air pockets, wrinkles, or foreign object contamination should be absent. Defects can require scrapping the entire laminate or reduced part strength. Wrinkles can cause delamination between layers. Another important factor in determining laminate quality is the resin distribution across the laminae. Resin-starved

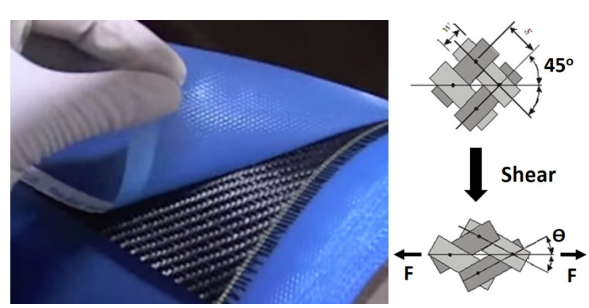

Fig. 3. Prepreg (carbon fiber infused with epoxy) covered with backing paper [2] (left), and an illustrative example of in-plane shearing mechanics (trellis shear) for woven fabric (right).

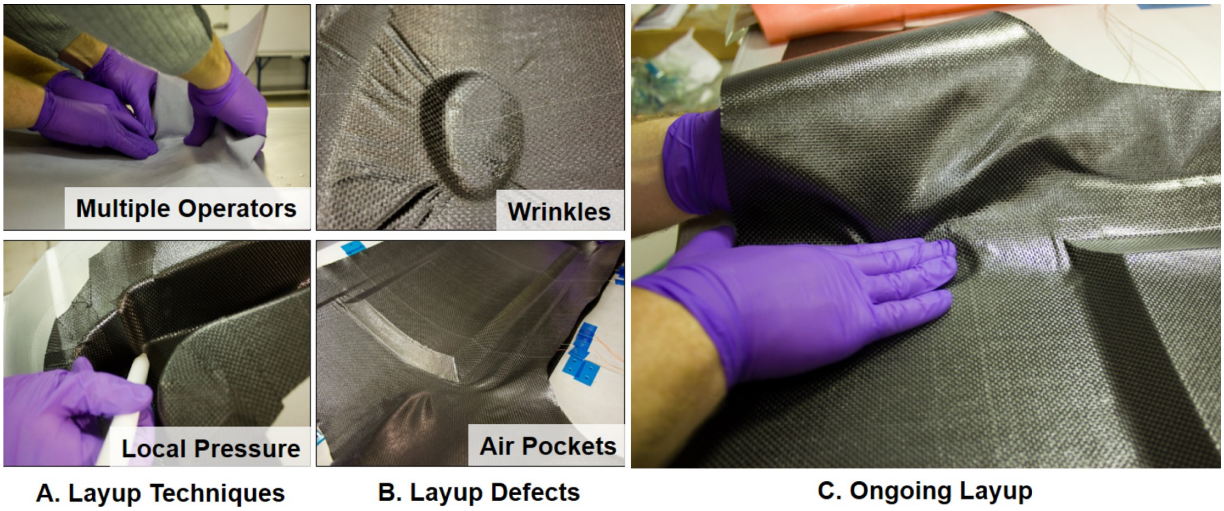

Fig. 4. (A) Layup techniques involving use of multiple operators and local pressure application [59]. (B) Layup defects like wrinkles and air pockets. [59]. (C) Example of an ongoing layup.

regions can cause the fibers to fail prematurely, as no matrix is present to distribute. Despite these challenges, hand layup is used for complex molds or production volumes where current automation solutions are technically or economically infeasible.

3.2. Draping and grasping robots

The cell consists of two types of robots, (a) the draping robot and (b) the grasping robot. The draping robot applies local pressure on the ply to conform it to the surface of the mold. The grasping robots manipulate the ply to facilitate the layup operation. The robots are equipped with custom end-effector tools that are designed as per the mold geometry. A RGBD-camera is used for monitoring the layup process during execution.

The draping robot needs to account for machining errors during the manufacturing of the mold and the registration errors while placing the mold in the work cell. This requires the robot to operate under impedance control. We have selected 7 DOF KUKA iiwa robot for draping that provides impedance controller. The controller allows the robot to comply with the surface irregularities as well as maintain constant force during the draping operation. Same robots are used for grasping as well. For larger and more complex geometries, the grasping robots require force feedback while manipulating the ply. Tension in the ply also must be accurately controlled, and the robots need to comply if the ply is being sheared or stretched too much. We can achieve these objectives under impedance control.

3.3. End-effectors

*Draping Tools:*A roller is used to replicate the motion of human hands to drape the ply. A cylindrical roller is effective for the conforming ply on single- and double-curvature convex regions of the mold. Fig. 5(A) shows the fabricated cylindrical roller. The main body is additively manufactured with ABS. The support structures for the roller are aluminum for reducing the dimensions to access compact features within a mold without making any unwanted contact with the ply. For concave regions with small radii, we have designed a conical roller Fig. 5(B), and a dibber Fig. 5(E) tool. The conical roller is used for improving the draping speed along the concave regions, but it poorly conforms plies to areas with tight corners. This limitation of the conical roller is overcome by the dibber tool, although it reduces the conforming speed. The end-effectors used in our work are inspired by the work done in [25].

Better conformity can be obtained by heating the prepreg. Heating improves tack and assists in adhering the ply to the mold. Heating also reduces the viscosity of the infused resin so fibers can deform easily and take shape. We have designed a heating system that applies heat locally to the region that will be conformed as the draping robot moves.*Grasping Tools:*Commercial grippers (Robotiq) are used for grasping. Prepreg readily adheres to the rubber fingers of the gripper. Teflon (PTFE) is appropriate for handling prepreg. We have designed custom gripper fingers to grasp and manipulate the ply. Along with it, internal channels are provided to carry compressed air. A high-velocity jet of air detaches the ply from the gripper fingers if it adheres to the gripper. The air jet is deployed to prevent the ply from being dragged with the grippers when they change location. Fig. 5(C and D) shows the gripper and the CAD model with internal channels used in the setup.

3.4. Part inspection

We assess layup quality using four metrics described as follows:

1. *Conformity:* The degree to which the sheet has conformed to the substrate. A map is generated to compare the point cloud of the layup with respect to a reference (CAD or substrate).
2. *Fiber Alignment:* Measure of the deviation of fibers in the woven fabric with respect to a reference or average alignment in a region.
3. *Resin Distribution:* Measure of distribution of resin (ideally homogeneous) throughout the fabric after the layup process.
4. *Other Defects:* Foreign object deposition (FOD), air pockets, wrinkles, or fiber damage.

The cell is equipped with Apodius Vision System 2D, Hexagon absolute arm with an integrated laser scanner, Intel RealSense D-415 depth camera, and DinoLite digital microscope. Fig. 6 shows these instruments in images A, B, C, and D respectively. We use the depth camera for monitoring the state of the sheet in real time. The state of the sheet is extracted after occluding the robots and the grippers from the cell. Online modification to grasp plans generated in this paper can be made by using feedback from the depth camera. Defects such as air pockets, and wrinkles can also be detected using the sheet state. Torque sensors in the robots monitor the force being applied at the end-effectors. Necessary compliance and local adjustments are made using the data from these sensors. DinoLite digital microscope is used to study the distribution of the resin in the layup.

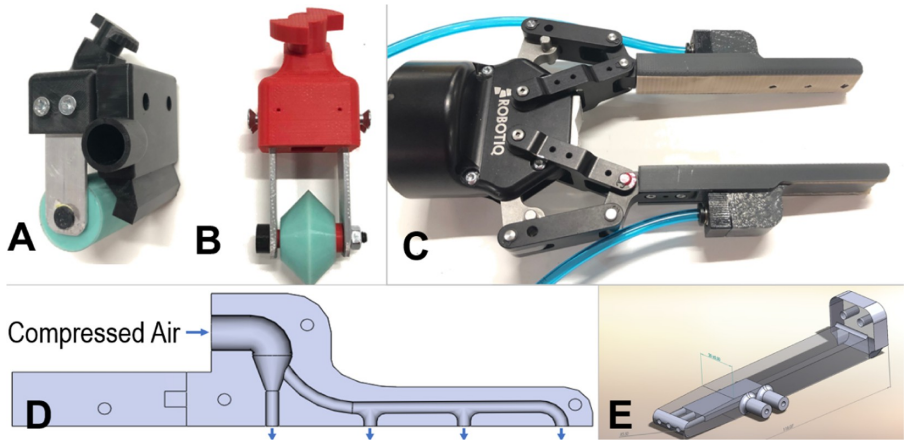

Fig. 5. (A) Fabricated cylindrical roller. (B) Fabricated conical roller. (C) The gripper used. (D) Section view of the gripper fingers showing the air channels. (E) CAD model of the dibber used in the process [62].

Fig. 6. Inspection equipment used in the robotic cell. (A) Apodius 2D sensor. (B) Hexagon absolute arm with integrated laser scanner. (C) DinoLite digital microscope. (D) Intel realsense D415 depth camera.

4. System architecture

Fig. 7 shows the system architecture of the robotic layup cell. The cell takes an expert input and automatically generates the plans for the robots. We categorize the automated composite layup process into three phases: (a) planning phase, (b) setup phase, and (c) execution phase. The expert user input is discussed in detail in Section 5. We will now discuss the three phases of our system.

4.1. Planning phase

The planning phase is responsible for taking the expert generated inputs, and converting them into actions that can be executed by the robots for performing layup operation. Expert user input provides the draping tool paths and some additional information that is used to generate the grasp plans and robot placements. The process parameters based on the mold geometry, grasp plans for manipulating the ply, robot placements with respect to the mold, and the draping and

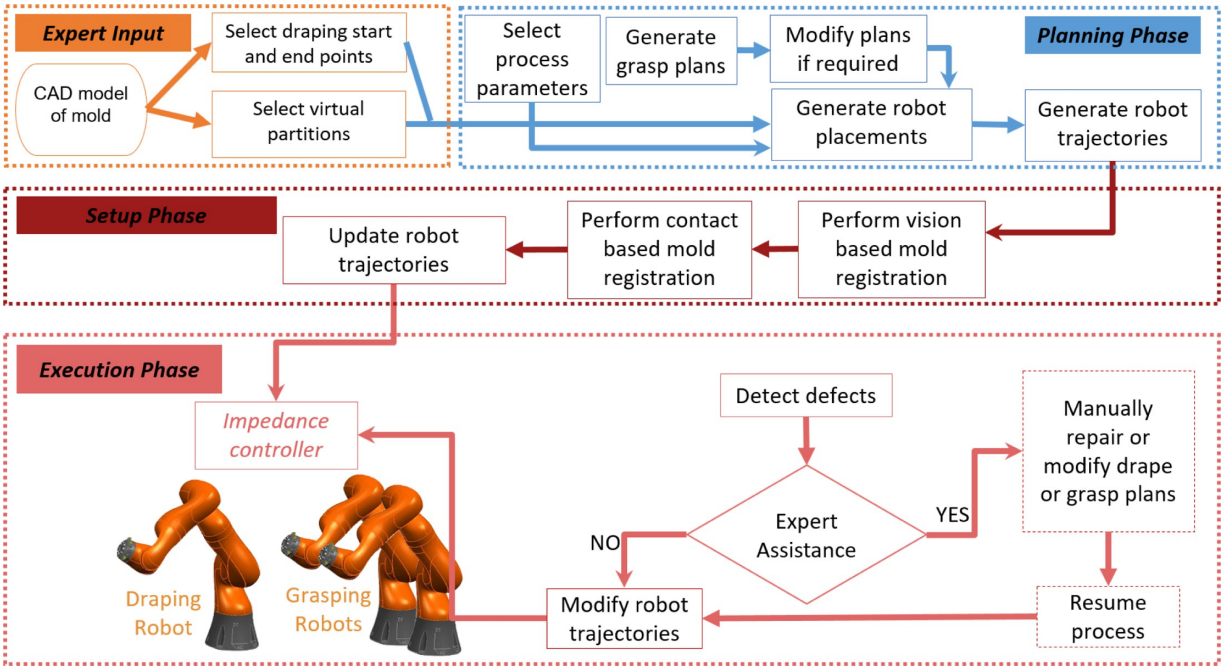

Fig. 7. System architecture diagram illustrating the three main phases in the robotic cell for composite layup.

grasping robot trajectories are computed in this phase. The grasp plans are visualized through an interface. The user can modify these plans through the interface, if required.

The system selects the process parameters required, given the CAD geometry and the type of material. We investigated the bounds on these process parameters by conducting offline experiments on common mold geometries [62]. For this work, the draping force (F_{drape}) and the dibbing force (F_{dibb}) will be bounded within [15,30] N. Maximum Cartesian velocity of the roller used will be 50 mm-s^{-1} and the temperature of airflow used will be 45° C. The stiffness of the end-effector under impedance control is bound within [1500,3000] N-m^{-1}.

4.2. Setup phase

The setup phase consists of all the components that are required before the physical layup process begins. This phase involves placing the robots at the computed locations with respect to each other and to the mold, performing a vision and contact-based mold registration, and recomputing the robot trajectories.

The robot placement algorithm will give the transformation of the robots with respect to the mold. The cell operator can then place the robots using this transformation. However, in practice, the exact transformation of the mold is different so the transformation error must be corrected. The CAD model provides a reference point cloud. We then generate a point cloud from the depth camera and compare it with the reference point cloud. An iterative process similar to iterative closest point (ICP) is used to find the transformation of the point cloud with respect to the robots. We then perform contact-based registration to obtain an accurate transformation of the mold. The new transformation is used to recompute the robot trajectories.

4.3. Execution phase

The robot trajectories are executed by the impedance controller. A Cartesian impedance controller measures the joint torques and computes the force at the end-effector to comply as per hooke's law. Sensors are used to monitor defects during the process. We use torque measurements and camera feedback for monitoring. The feedback is used to refine the drape and grasp plans to prevent the defects. An expert is called for assistance if the robots cannot repair the defects. The human operator also performs minor adjustments to the grasp plans if required to prevent defects from occurring. An online grasp plan modification strategy has also been presented in our work in [58].

5. Getting expert user input

Expert input is a part of the initial setup process which simplifies the grasp planning problem. The user selects regions on the mold which deconflicts the workspaces of the robots. Grasp points which can be used by the robots are also specified. More details on how expert input is used in planning is described in Section 6.5. A drape simulator is used to simulate how a fully draped ply appears on the mold. Fig. 8 shows the molds we are using to determine the grasp plans and the corresponding drape states simulated. The expert can estimate the drape sequence to be followed to drape the ply from these simulations. We use a virtual fiber placement simulator or VFP provided by the research group at the University of Bristol.

The robots must follow the drape sequence. We have developed an interface - the human operator selects the mesh triangles that belong to each discrete stage. Fig. 10 shows a few of the discrete stages in the draping sequence for one of the molds used for evaluation. Stage 1 in the figure shows that the operator starts by adhering to the ply in the center portion of the mold, which is also the highest point on the mold. Later, the operator proceeds downwards towards the right in stage 5, and finishes one half of the mold by stage 10. Finally, the left half of the mold requires four more discrete steps to cover the entire mold surface. Fig. 9 shows the draping regions selected by the technician for all the molds that are used to evaluate the layup operation. Each region in the figure is shown in a distinct color and labeled as *RX*, where *X* is the number of the region. For example, Region 1 is denoted by *R*1.

In each region *RX* of the draping sequence, the robot draping tool must make contact with the ply and apply the necessary force to attach the ply to the mold. In this paper, we use the technician's assistance to provide the path that the draping tool should follow while laying up the ply. We have developed an interface (see Fig. 11) which the technicians can use to input the draping paths. The user can select the start and endpoints of each drape path on the mold surface, and the software generates geodesic curves on the mold that connect these points. These geodesic curves comprise discrete waypoints of the surface of the mold, and we term this collection of discrete waypoints as draping tool paths, which are used to compute draping robot trajectories.

During the layup process, the grasping robots are used to manipulate the ply before the draping robot adheres the ply to the mold. Given the size of the ply, the number of robots used for grasping the ply varies. In the current version for generating the grasp plans, we deconflict the regions on the ply that must be handled by each robot using a virtual partition. These virtual partition lines are provided by the

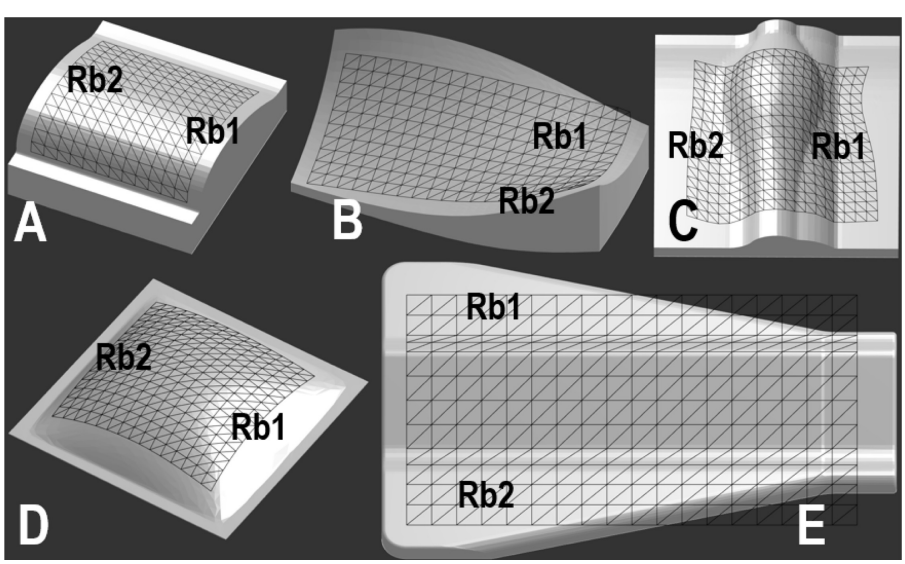

Fig. 8. The molds used in our work. Fully draped mesh obtained from virtual fiber placement is shown on the molds.

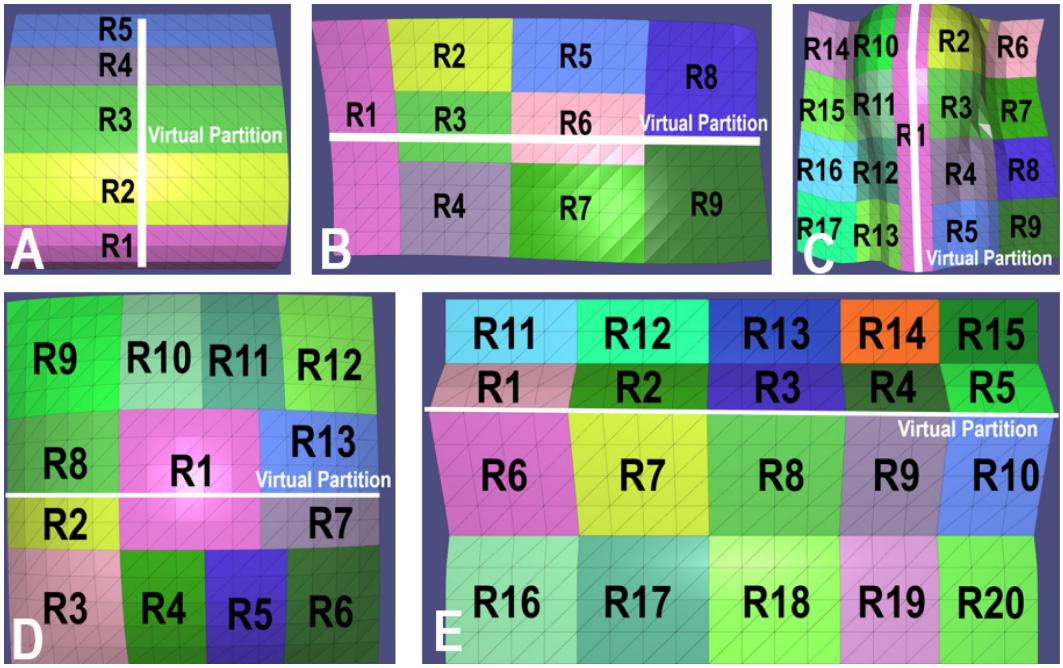

Fig. 9. The user interface is used to select region (R) in the draping sequence, virtual partitions, and mesh triangles along the ply edges which the robot can use to grasp the ply.

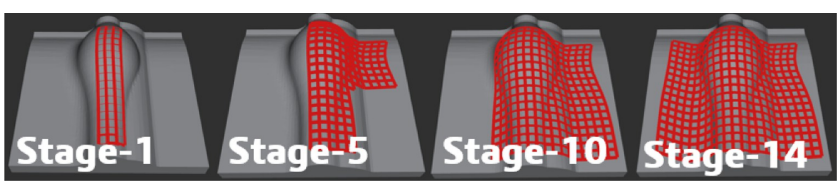

Fig. 10. Four intermediary steps of the draping sequence are shown. In all the draping sequence is discretized into 14 steps. Step-14 is the fully draped ply. Unconformed ply is not simulated.

technicians for all the mold and shown by the white center line in Fig. 9. Section 6.4 provides more details about the partitioning and its use in the grasp planner.

6. Grasp planning

6.1. Background

The grasping robots are responsible for manipulating the ply during the draping process. If accidental contact is made between prepreg layers, air entrapment is possible. Ply collision with the mold must be checked in order to prevent contact. Alignment of the ply must be maintain with a specific coordinate axis on the mold. Misalignment of the ply can be caused if the ply excessively bends in a region, and the draping tool moves over the region. Excess tension or bending in the ply are high deformation states that can cause wrinkles. Such high deformation states can be characterized by an increase in the energy of the ply. Thus, we must generate the plans which keep the energy of the ply low.

The ply is represented by a mesh. Fig. 12 illustrates a mesh over the

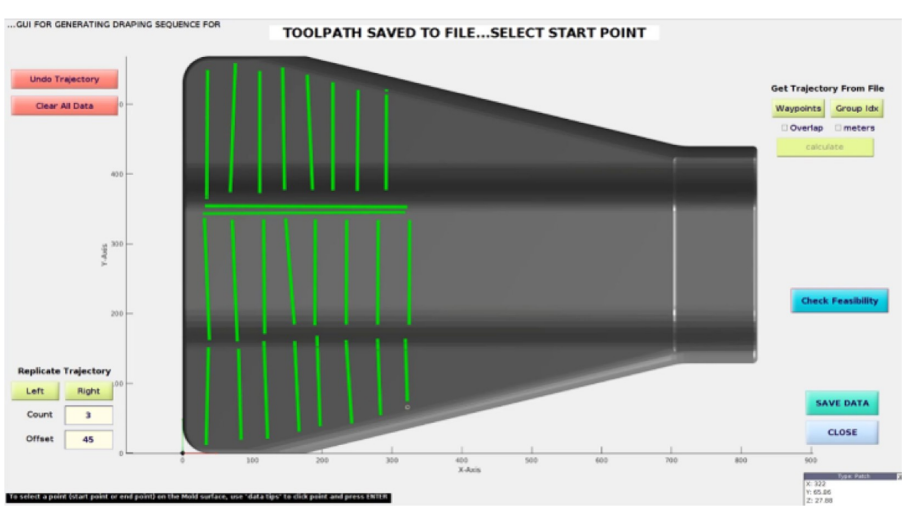

Fig. 11. The graphical user interface (GUI) used for the selection of draping tool paths is shown. The CAD model and a draped mesh approximating the ply is also illustrated.

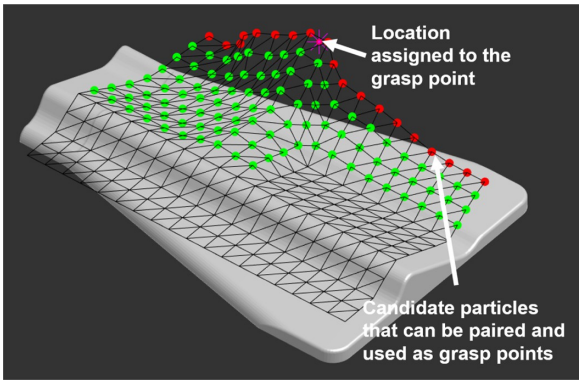

Fig. 12. Grasp points along the edge of the mesh are used by the robots for grasping. The state of the ply simulated while holding it from a grasp point at a location in space is also illustrated.

mold. We represent a grasp point as the center of a pair of consecutive vertices along the edge of the mesh. The robots can hold the ply from a specific grasp point along the edge of the ply. We can determine the combination of grasp points and their a location in space to obtain a state which is favorable to satisfying some constraints for a given conformed region. Fig. 12 illustrates different grasp points along the edge of the ply and ply state by holding it at a grasp location from a specific grasp point. In this work, our planner only considers the 3D coordinates along X, Y, and Z axes in Cartesian space for assigning locations.

In the proposed framework, we have discretized the draping process into a sequence of regions. As stated earlier, this sequence of regions is given by a human expert. Each region with conformed and unconformed ply vertices in it is termed a *stage*. Fig. 10 shows only the conformed vertices of the mesh for some stages in the discrete sequence on a mold. Unconformed vertices are not shown, as their state is determined by where the ply is being held. In total, there were 14 discrete stages selected in the drape sequence. The figure illustrates four intermediate stages. Stage-14 is the fully draped ply. We can also increase the resolution between stages by discretizing them with smaller areas, which will result in a smoother solution but at the cost of more planning time.

The sequence of grasping points and their locations for every stage in the process defines the *grasp sequence*. If the grasp points are different for consecutive stages, a re-grasp motion is required. If the grasp point remains the same between the stages, but the locations are different, a repositioning motion is required. Fig. 13 illustrates this concept between two consecutive stages. There are two robots holding the unconformed ply, which is represented in green. One of the robots has to grasp from a different grasp point, so a regrasp motion is required. The other robot had to reposition the ply from the same grasp point. The sequence of grasp points are computed by the grasp planner. Deformation of the sheet during regrasping motion is not accounted for during offline computation. An online refinement strategy is implemented for considering the deformations that happen when the sheet

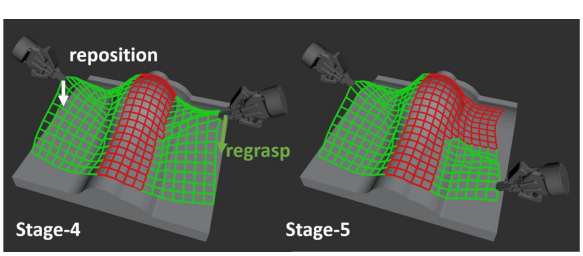

Fig. 13. Regrasp and repositioning concepts are illustrated for two consecutive stages.

is released [58]. The draping robot waits for regrasping motion(s) (if any) to be completed between stages. However, trajectories are continuously executed to conform to the area that is represented by a stage.

We present a solution framework by posing the grasp sequence generation problem as an optimization problem with constraints. Expert users seem to prefer certain types of sequences. These preferences are validated by conducting experiments. User preferences are modeled as a cost function, developed as a part of our approach. Different components of cost function describe aspects of user preferences. We use a weighted cost to capture different aspects of preferences. A grasp sequence that satisfies all constraints and minimizes the given cost is the solution.

6.2. Problem formulation

We assume the CAD $\mathcal{M}$ model of the mold is given. $\mathcal{M}_v$ represents the voxelized map of the CAD model. The set of parameters (material properties and scaling constants) used in the drape simulator is represented by $\vec{P}$. Consider a draping sequence $\mathcal{W}_d = \{W_1, W_2, ..., W_r\}$, where W_i is a stage and r is the total number of discrete stages in the draping sequence. A set of available grasp points along the edge of the unconformed ply is computed. For every stage, we will have this set given by $G_r = \{g_1, g_2, ..., g_k\}$. For an unconformed ply, all the grasp points will be available, in which case k equals the total number of grasp points on the ply. If we assign a location $< x, y, z >$ in the Cartesian space to the grasp point, we will be able to simulate the state of the ply. We denote this *location assignment* as $l(g_k)$ to the kth grasp point. A grasp sequence can be represented as $G_j = \{l(g_{k1}), l(g_{k2}), ..., l(g_{kr})\}$ which is essentially a sequence of assignments of location to the corresponding grasp point for every stage $W_i \in \mathcal{W}_d$. The grasp planning problem is formulated as finding a grasp sequence that minimizes a cost. Eq. (1) defines the grasp planning problem.

$$G_s \leftarrow \arg\min_{G_i} C_t(G_i, \mathcal{W}_d, \mathcal{M}_v, \vec{P}) \tag{1}$$

G_s is found by minimizing a cost function that has three main components, as discussed earlier. The next section defines the cost function.

6.3. Cost function components for guiding grasp planning

Energy Function: Breen et al. used a physics-based simulation to generate the grasp plan, and our approach is similar to their work [11,12]. Each particle (vertex) is allowed to translate along the X, Y, and Z axes in 3D Cartesian space. The material properties are encoded as an inter-particle relationship. Fig. 14 shows a particle P_i surrounded by its manhattan neighbors. Deformed and undeformed configurations of the mesh surrounding the particle are shown. The link between a particle under consideration and a neighbor is represented by a spring. The mass of the ply is divided amongst all the particles. This mass-

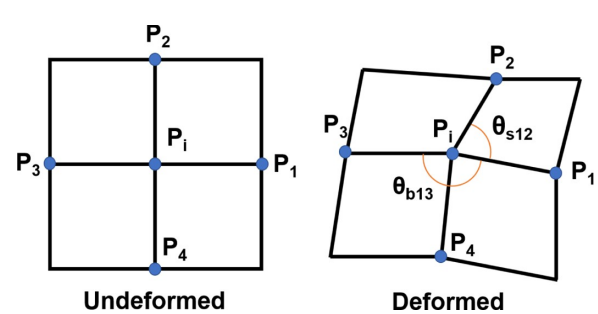

Fig. 14. Neighborhood of a particle P_i in the mesh. Illustrations of the deformed and undeformed mesh are provided. The position of the particles and shear and bending angles are used to compute the potential energies.

spring model is widely used for simulations in the computer graphics community. The physical state of the ply represented by n particles in the 3D Cartesian space is defined by the $3n \times 1$ coordinate vector $\vec{X} = <x_1, x_2, x_3, \cdots, x_{3n}>$.

At any state, the total energy of the ply is represented by the following four potential energies: spring, bending, shear, and gravitational. A particle pair is connected by a linear spring in our model, so the spring potential is given by $k*dx^2$, where dx is the change in the distance between the particles. We define the shear and bending energies with respect to the radian angle between the concerned particle, and its neighbor. Fig. 14 shows the shear angle between particle P_i and its immediate neighbor P_1. Three other shear angles between P_i and the neighbors P_2, P_3, and , P_4 can be computed. Bending energy is defined between the pairs P_i, P_1 and P_i, P_3. We conducted two different tests to determine the relationship between shear and bending energies and the corresponding radian angles of deformation. The tests and the relationships obtained are discussed in Section 6.7.2. The gravitational potential is represented by $m*g*h$, where m is the mass of the particle, g is the gravitational acceleration, and h is the change in height (z coordinate of the particle) relative to the mold.

The state of the ply under given boundary conditions is found by minimizing the overall energy of the ply. The boundary conditions in our case include restricting the particles in the draped region and those gripped by the robot from any motion. The rest of the particles are free to move. We are interested in finding the lowest energy state, which will be the solution to the simulation. The unconstrained optimization problem given by the Eq. (2) is solved to find the solution. We use the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) [56,76] algorithm from the nlopt library [41] to solve the problem.

$$\vec{X} \leftarrow \arg\min_{\vec{X}} \sum_{i=1}^{n} (U^i_{spring} + U^i_{shear} + U^i_{bending} + U^i_{gravity}) \tag{2}$$

Deformation energy cost can also incorporate misalignment constraint for the ply. For instance, if an edge of the ply needs to be aligned with edge of the mold, the cost will be higher if the edge excessively bends and misaligns. The grasp point and location will be biased towards such regions in order to keep it relatively flat so that alignment is maintained. Alignment costs can also be incorporated as separate inequality constraint over grasp point and location where robot is forced to grasp from predefined points and stay within location tolerances. Such constraints are often defined by the process itself.

Regrasping Cost Function: When the robot needs to *regrasp* the ply, the gripper will open and release the ply, slide along the edge to the next grasp location, and close. The cost function used in our approach minimizes the number of regrasps required for the process. We use the Euclidean norm of the distance between two consecutive grasp locations of the respective grasp points to find the regrasping cost.

Collision Cost: We use the Euclidean distance transform (EDT) of the mold to detect collisions. The position of each particle in the unconformed region of the mesh is used to query the EDT map. The collision has a binary value. If the EDT value is below a certain threshold, the particle in the ply collides and vice versa. The threshold value for declaring if a particle is colliding is the same in case of uniform collision cost. Different threshold values for every particle are used for a non-uniform collision cost. For instance, draped, to-be draped, and free regions on the sheet can be assigned different collision costs for ease of implementing feedback systems in the future.

Weighted Cost: In order to maintain the quality of the layup, we should minimize the area under the absolute energy vs. stages curve. Collision and regrasping costs are then incorporated into the cost function along with the integration of the absolute energy over stages. This cost function C_t is given by the Eq. (3). C_u, C_c and C_g are the energy cost, collision cost, and regrasp cost respectively. We measure collisions using $\mathcal{M}_v$ and the regrasp cost as the euclidean distance between two expanded node states $l(g_{k1})$ and $l(g_{k2})$ for two consecutive stages.

$$\begin{aligned} C_t &= w_1{}^* \int_{W_r}^{W_i} C_U + w_2{}^* \sum_{W_r}^{W_i} C_c + w_3{}^* \sum_{W_r}^{W_i} C_g, \\ C_u &= U^*ds \\ C_c &= n_c/n_f \\ C_g &= \|l(g_{k1}), l(g_{k2})\|_2 \end{aligned} \tag{3}$$

where n_c is the number of colliding particles and n_f is the number of free particles for that stage. Also, $w_1 + w_2 + w_3 = 1$. U is the absolute energy of the ply and ds is the normalized value representing a stage. For instance, if the process has r discrete steps, $ds = 1/r$.

6.4. Virtual partitioning of state space

We can use any number of robots for manipulating the ply. In the current version for generating the grasp plans, we deconflict the regions on the ply that needs to be handled by a single robot using a virtual partition. Fig. 9 illustrates this concept for our case. The user selects the regions which will be handled by the corresponding robots. For two robots, we represent the two regions separated by a white line. The corresponding robot number is labeled in the figure. Each of the regions is termed as a virtual partition. Usually, in a draping process, the ply is conformed along a line path that runs on the mold. The path inherently deconflicts the workspaces for the molds. In Fig. 9, for the sequence shown on mold C and E, the first conformed region R1 will split the robot workspaces in two without conflicts. However, for other molds, both the robots will have shared unconformed regions as the free regions for both the robots interact with each other. A robot action in conflicting regions will influence the grasp location of another robot as an unconformed ply state is being shared between them.

Even though the robots will have shared unconformed regions, we solve the grasp planning problem for each robot separately. We find nominal grasp sequences for each robot. We then use these nominal grasp sequences to account for the conflicting regions. Section 6.6 discusses the algorithm in detail. In our case, we use two robots. We solve two independent, grasp planning problems. Fig. 15 illustrates the use of virtual partitions in our approach. The stages and grasp points available for grasping in the case of mold A for the robot-2 are shown in the figure. While planning for robot-2, all the particles that belong to the virtual partition of the robot-1 remain in their draped position.

6.5. State space representation

The available grasp points are known beforehand. We can construct the graph $\mathcal{G}$ with the nodes as the available grasp points in every stage. In our approach, selecting a node in the graph is equivalent to selecting a grasp point. A grasp point is selected and then moved around in a discrete Cartesian space. Cost is evaluated at each discrete location. A location that reduces the total cost of the grasp sequence is assigned to the grasp point for that stage. We call this a location assignment search. The location assignment search is discussed in detail later in this section. We evaluate the node cost by using the location assignment search.

We use an approach based on a backward simulation, where we start with a fully draped ply. The root node corresponds to the fully draped ply stage, where no grasp point is available as we progress along with the discrete set of stages; the ply unconforms until it is fully unconformed. This framework can be used to propagate information into later stages. The graph is an acyclic directed graph. Fig. 15 illustrates the different stages in the process. Unconformed ply particles are highlighted in green. The corresponding graph $\mathcal{G}$ is also shown. As the stages progress, the grasp points that are available to be used are represented by integers along the edge of the ply. G_i essentially is a grasp sequence of nodes, forming a path from the root to the leaf node having the lowest cost. The figure also shows one such path in the graph highlighted by forwarding arrows. As the graph is acyclic and directed,

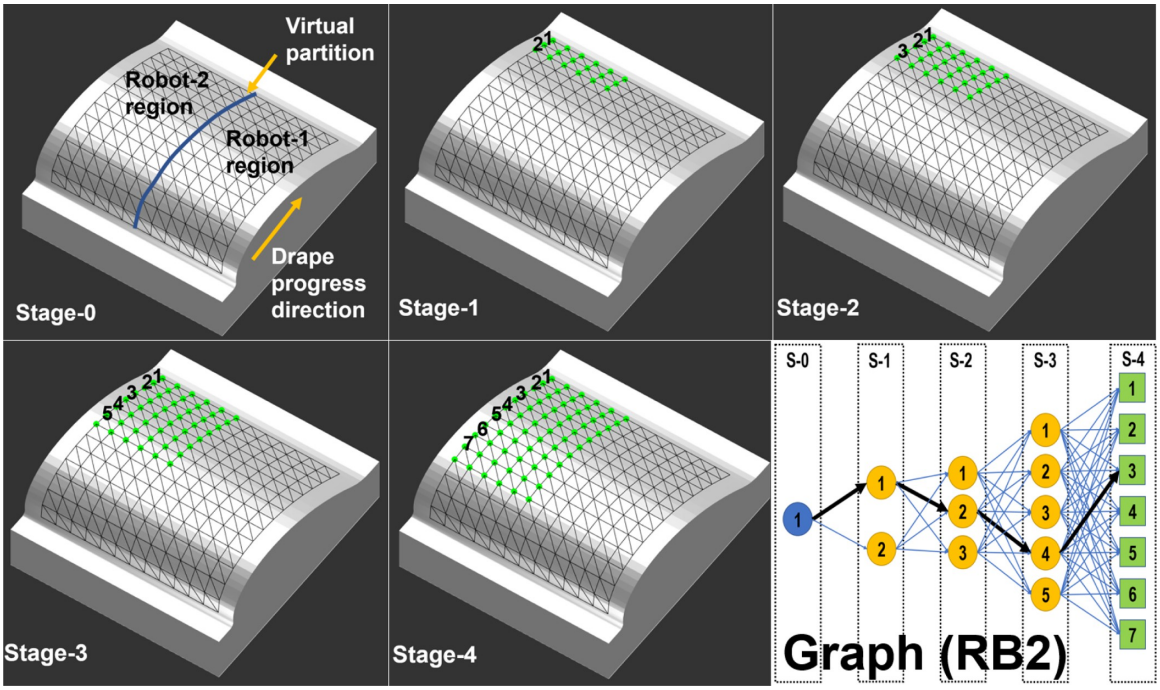

Fig. 15. Different stages in the backward tracking graph are illustrated. The available grasp points are marked by integers along the edges of the ply. Virtual partition generated by the user deconflicts the regions handled by the robots. Separate graphs are generated for the robots. The graph generated for the robot 2 (RB2) is shown here.

minimum cost grasp points in every stage lead to an optimum grasp sequence.

A location assignment search is conducted by varying the height (Z-axis) of the grasp point and moving it forward or backward (Y-axis) relative to the mold. Fig. 16 illustrates the Z and Y axes of the motion for a grasp point. We discretize the motion along these axes. Motion along the edge of the ply (X-axis) is equivalent to changing the grasp point. The cost varies as a grasp point is moved during the location assignment search. Out of all the different locations that can be assigned to a grasp point, we pick the one with the minimum cost. A state space search is then used to find the optimum solution. In our implementation, Gradient descent algorithm is used to find the minimum cost location. The initial location assignment is given by a heuristic function, which also reduces the computation cost. The heuristic used is described in detail in Section 6.6.

The local minimum found by an optimization algorithm, such as gradient descent, is the global minimum due to the behavior of the cost function in this application domain. Fig. 16 shows the behavior of the energy and collision costs as a function of the Z location of a grasp point. Collision cost decreases as the ply is lifted, but energy cost starts increasing due to high deformation and potential. At one point, a steep increase in energy is obtained with less reduction in a collision. Similar behavior is obtained with these costs when the ply is lowered.

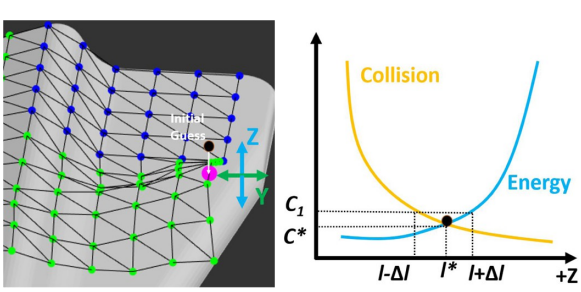

Fig. 16. (Left) Optimum grasp location (magenta) obtained using an initial guess (black) in the local assignment search. Z and Y axes along which the state space is discretized are shown. (Right) Behavior of collision and energy in the local neighborhood of optimum grasp location. Global minimum (l^*) is found by a gradient based algorithm within the domain. Any step Δl leads to an increase in the cost. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.6. Algorithm for generating grasp plan

Our algorithm is an any-time algorithm. It generates a grasp plan and then updates it further to reduce the cost. The algorithm can be terminated if a time limit is reached, the cost cannot be decreased any further, or all the nodes have been evaluated.

Our goal is to use domain-dependent heuristics to generate a low-cost initial plan and obtain an optimal resolution solution in a computationally tractable time. We propose an ordering heuristic that prioritizes the nodes in every stage. The ordering heuristic first explores the state space to prioritize the nodes (grasp points) which are promising. The planner begins by exploring a small number of grasp locations for every node. Two grasp locations which are equally spaced between the mold surface and a maximum height above the mold are evaluated. The location with a lower cost is picked and assigned to the corresponding grasp point. All the grasp points in a stage are then ordered based on the respective assigned costs. The procedure is represented as *PrioritizeNodes* in Algorithm 1. We then use the generated lower cost grasp locations as an initial guess for the gradient descent step used in local assignment search. The heuristic significantly reduces the computational cost. It allows us to generate good initial guess for the gradient descent search during location assignment. A minimum cost solution is then found with fewer iterations as opposed to exploring the entire search space for the optimal solution. Additionally, prioritizing nodes helps us evaluate the grasp points which can have a lower cost in the beginning of the algorithm. As a consequence, our algorithm generates grasp plans with a low initial effort compared to randomly selecting nodes.

The planner goes through all the stages and evaluate the nodes which have the highest priority as determined by the ordering heuristic in the first iteration. The first iteration provides an initial grasp plan

1: $\mathcal{G} \leftarrow GetGrspPts(\mathcal{W}_d)$
2: $PriorQ \leftarrow PrioritizeNodes(\mathcal{W}_d, \mathcal{M}_v, \vec{P})$
3: $path \leftarrow GenInitialPath(\mathcal{G}, PriorQ)$
4: $iter \leftarrow 0$
5: $boolList \leftarrow Identity(sizeof(\mathcal{W}))$
6: **while** $UntilConvergence$ **do**
7: **for** $s \in [0, \mathcal{W}]$ **do**
8: **if** isZero(binList) **then**
9: $binList \leftarrow Identity(sizeof(\mathcal{W}))$
10: **end if**
11: **if** binList(s) **then**
12: $path \leftarrow UpdatePath(\mathcal{G}, \mathcal{W}, PriorQ, \vec{P})$
13: $PriorQ \leftarrow UpdateOrder(PriorQ, path, \mathcal{G}, \mathcal{W})$
14: **end if**
15: **end for**
16: **end while**

Algorithm 1. GenGraspPlan($\mathcal{W}_d$, $\mathcal{M}_v$, $\vec{P}$).

having a low cost. In the successive iterations, steps 6 to 16 of the Algorithm 1 are followed. We maintain a list of boolean values (1 or 0) corresponding to each stage. All the members in the boolean list are initialized to one. If a stage showed improvement in the previous iteration (boolean value 1), Another node is evaluated from the top of the prioritized queue. The path is updated (*UpdateOrder*) only if the cost of the new path is less than the previous cost. The newly evaluated node provides information to reprioritize the order of the nodes. We can update this order using the new information at *UpdateOrder*. If the stage does not show any improvement, a node is not evaluated in the next iteration from that stage. The boolean value corresponding to the stage is set in the list to zero. When all the members in the boolean list are zero, and the stopping condition has not been met, we reinitialize all the elements to one (lines 8 - 10 of Algorithm 1). This allows us to bias the search towards evaluating more nodes from the stages that are showing a reduction in cost.

This algorithm is used to find solutions for the grasp plans in each of the two-state spaces corresponding to each robot separately in our case. Consider the mold A shown in Fig. 17 as an illustration. The two independent solutions are found for robot-1 (RB1) and robot-2 (RB2). The first two images show that only the unconformed particles in a robot's virtual partition are considered during evaluation. The unconformed particles in the other robot's partition are considered to be fixed at their draped position during the simulation. The third image in the sequence shows the combined solution, which is generated from the nominal grasp locations. The ply collides with the mold so a state space search is used to reposition the grippers of both the robots in order to minimize the cost. Gradient descent is used to adjust for the grasp locations, which gives us a solution shown in the fourth image.

Table 1
Dimensions, length of each link in undeformed mesh, and number of stages excluding the stage-0 are provided.

Dimensions (mm)				Link length in undeformed mesh (mm)	Number of stages excluding stage-0
Mold	Length	Width	Height		
A	500	450	200	32	4
B	440	220	140	20	8
C	570	570	110	25	16
D	300	300	56	15	12
E	820	570	75	40	19

6.7. Computational results

6.7.1. Test cases

We have tested the grasp planner on five different mold geometries. Fig. 8 shows these molds and the fully draped plies on them. The molds are inspired by industrial use cases. They cover the basic geometries like single curvature to more complex concave and double curvature geometries. We do not use the entire mold for draping the ply. A portion of the mold around the perimeter is used for vacuum bagging, which is a part of the curing process. These unused regions are visible on the molds. The dimensions, link length, and the number of stages for each mold are provided in Table 1.

6.7.2. Drape simulator

We will first discuss the results from the drape simulator being used for determining the unconformed state of the ply during the process. Shear and bending energy relationships to be used for the simulation are determined. For shear, we have used the bias extension test data, which generates the force (F) vs. deflection (dS) curves. The shear energy relations is found using the equation $U_{shear} = \int FdS$. Bending energy, on the other hand, is found by using the beam theory. Different samples of prepreg are used to find the 1-dimensional bending under its own weight. The bending energy for a cantilever beam is given by the equation $U_{bending} = FL^3/(6EI)$, where F, L, E, and I are the force applied, lever arm, young's modulus, and the moment of inertia respectively. We determine the bending angle at discrete points along the length of the sample and compute the energy stored for that point. This gives us the bending energy relationship. Exponential function is used to best-fit the data collected from the experiments to estimate these relations. The shear and bending energy relationships with respect to the respective radian angles are given in equation 4 and (5) respectively.

$$U_{shear} = 9.132 * e^{(-4.485 * \theta_{shear})} \tag{4}$$

$$U_{bending} = 1.124e10 * e^{(-15.34 * \theta_{bending})} + 0.06109 * e^{(-1.1386 * \theta_{bending})} \tag{5}$$

The spring potential energy is adjusted to simulate the ply as closely as possible for different test cases under various conditions of grasping.

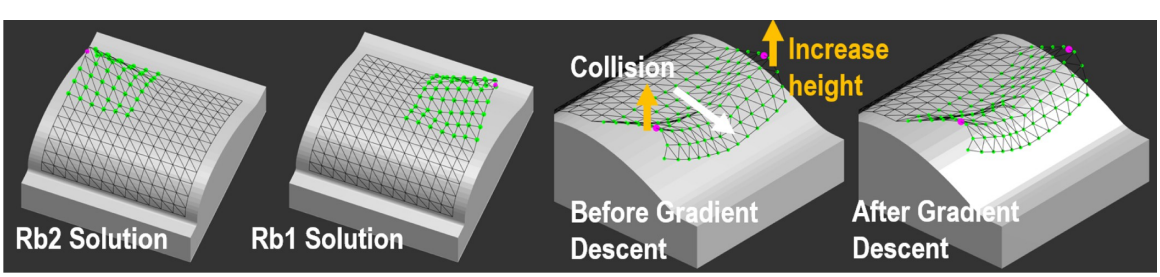

Fig. 17. (Left to Right) Independent solutions obtained for robot-1 (RB1) and robot-2 (RB2), ply state simulated for combined solution obtained from the nominal grasp locations, solution improved using a state space search using the nominal grasp locations.

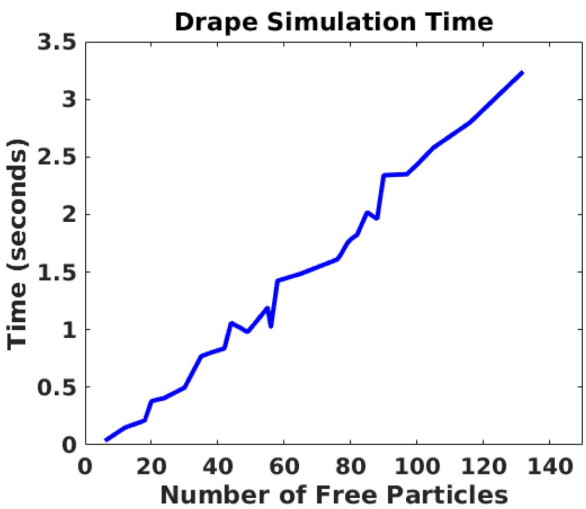

Fig. 18. The simulation time (seconds) for one simulation run vs the number of free particles in the unconformed region of the ply.

Plies used to tune the simulator are 250 × 250 mm and 500 × 250 mm in size. The current version of the simulator is implemented in C++ on a computer with an Intel Xeon 3.50GHz processor and 32GB of RAM. Fig. 18 shows the computation time for producing a solution state of the ply relative to the number of free particles in the ply. We can observe a linear increment in the computation time for the simulation with respect to the number of free particles. As numerical gradients are used in the current version of the implementation, the simulation times are high. This leads to high planning times. With analytical gradients and an adaptive mesh-based model, we can significantly reduce the simulation time.

6.7.3. Computational performance of grasp planner:

We benchmark the grasp planner with a brute force approach where all the nodes are evaluated to find the optimum solution. The time taken for evaluating each node is recorded by taking the average time for all simulation runs in the local assignment search. We present the number of nodes evaluated and the computation time taken for both the robots. Table 2 shows these results. The brute force approach is labeled as the baseline approach we are benchmarking against. The number of nodes reported is the total number of grasp points present in all the stages in the graph. MATLAB was used for the implementation of the grasp planner. We used parallel processing with 4 cores during the local assignment search to explore the discrete state space, which reduced the overall computation time. The location assignment search essentially uses drape simulations.

We compared the cost of the initial grasp plan generated by using the ordering heuristic with a randomly generated path. Table 3 shows the comparison. We can observe that the initial grasp plan costs are lower for our approach as compared to randomly generated paths. The initial grasp plan costs are also closer to the optimum costs, as reported in the table. The heuristic can provide initial plans that are closer to the optimum, which further reduces the computation time to find the optimum solution.

Fig. 19 shows the solution obtained for mold C. As we mentioned earlier, the stages are numbered in ascending order from the root node. The root node is the fully draped ply in a backward tracking search. We report the solution obtained in simulation for both the robots in the order of the drape sequence. Particles in the conformed regions are highlighted in blue, unconformed ones in green, and the grasp point is highlighted by a magenta marker. We can see the regrasps that occur during the process.

6.7.4. Scalability

The molds we have used range from smaller to medium-sized geometries. For instance, mold A has only 4 stages in the graph excluding stage-0 with a total number of 22 nodes. Mold E, on the other hand, has 19 stages and 300 nodes for robot-1. However, the computation time reduces as compared to the brute force approach by around the same factor for these molds. This shows that if we increase the size of the molds to larger-scale dimensions of over 2 m, our approach will still evaluate a fewer number of nodes for generating the plans.

We can also have the same number of grasp points, even with a higher mesh density. This is because the gripper will physically hold the ply using a 10–15 mm edge length. So even if the cell size in the mesh is 5 mm, we can combine more particles instead of two particles as used in our approach to create a grasp point. However, as the number of grasp points increases, the computation time increases.

We will now discuss the second important planning problem to automate the composite layup process. Once the drape and grasp tool paths are computed, we need to ensure that the robots are reachable and satisfy all constraints. The robot placement algorithm finds suitable placements as a solution to the problem.

7. Algorithm for robot placement

7.1. Background

The position of a body is defined by the $< x, y, z >$ components along the X, Y, and Z axes of a Cartesian frame. A quaternion vector

Table 2
Comparison of the baseline approach with our approach. Results reported are for robot-1 and robot-2.

Computation time (seconds) and Nodes evaluated										
	Robot-1 (RB1)					Robot-2 (RB2)				
	Baseline Approach		Our Approach			Baseline Approach		Our Approach		
Mold	Nodes evaluated	Time taken	Nodes evaluated	Time taken	% Reduction in Time	Nodes evaluated	Time taken	Nodes evaluated	Time taken	% Reduction in Time
A	22.00	43.10	7.00	4.17	90.32	22.00	35.45	5.00	2.60	92.67
B	50.00	164.76	14.00	17.90	89.14	73.00	223.09	47.00	61.67	72.36
C	102.00	742.11	25.00	113.89	84.65	230.00	2041.40	66.00	386.83	81.05
D	94.00	479.35	24.00	52.51	89.05	154.00	762.38	81.00	185.49	75.67
E	300.00	695.21	193.00	211.12	69.63	210.00	785.49	76.00	125.94	83.97

Table 3
Comparison of the cost of initial grasp plan for using ordering heuristic and random generation. The percentage reduction in the initial path cost when using ordering heuristic for both the robots is provided.

Comparison of Cost of Initial Grasp Plan								
	Robot - 1				Robot - 2			
Mold	Randomly Generated	Heuristic Based	Optimal Path Cost	Percentage Improvement	Randomly Generated	Heuristic Based	Optimal Path Cost	Percentage Improvement
A	0.36	0.10	0.10	72.10	0.38	0.13	0.13	65.69
B	0.71	0.20	0.19	72.30	1.42	0.87	0.75	38.96
C	0.85	0.64	0.33	23.73	1.60	0.93	0.92	41.45
D	0.67	0.34	0.31	49.12	1.37	0.84	0.62	39.12
E	2.31	1.41	1.29	38.83	1.32	0.91	0.86	31.34

$< q0, q1, q2, q3 >$ is used to describe the orientation of a body. Rigid body motions are described by attaching a frame to it. A frame is essentially defined by Cartesian coordinates to represent its position and unit direction vectors *bx, by, bz* along the X, Y, and Z axes, respectively. These unit direction vectors can also be used to obtain the orientation of the frame with respect to a reference frame. In this work, we will use the color scheme red, green, and blue to illustrate the unit vectors *bx, by, bz* for a frame. We use a homogeneous transformation ${}^{A}T_{B}$ to represent a frame B with respect to another frame A [17].

The CAD model of the mold is used to generate the tool paths for the draping and the grasping robot. The set $\mathcal{S}_{\mathcal{D}}$ and $\mathcal{S}_{\mathcal{G}}$ comprise of all the tool paths for the draping and the grasping robot respectively, where $\mathcal{S}_{\mathcal{D}} = \{S_1, S_2, \cdots, S_d\}$ and $\mathcal{S}_{\mathcal{G}} = \{S_1, S_2, \cdots, S_g\}$. S_d and S_g are represented as a set of waypoints. For simplicity, we describe a kth path using a set

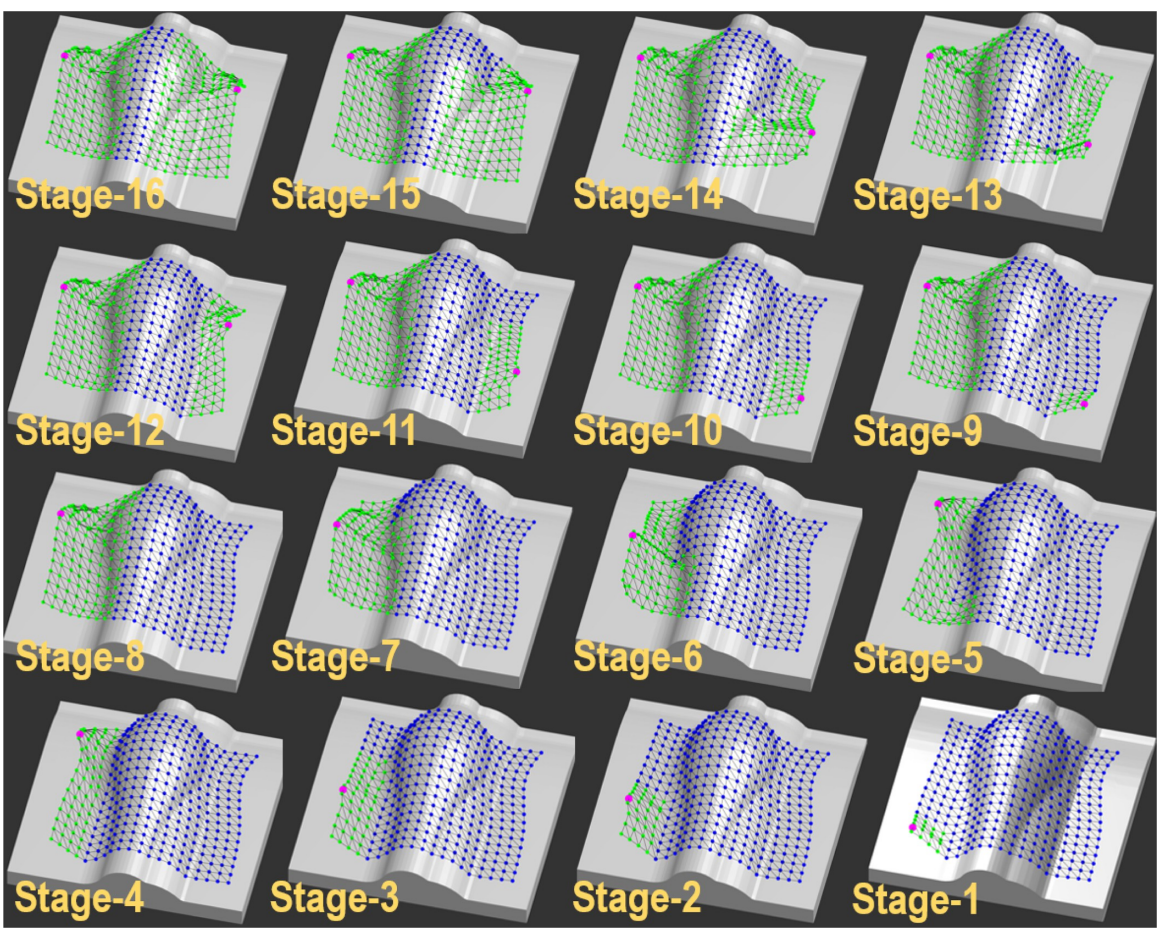

Fig. 19. Simulation of the solution obtained for mold C. Particles in blue are part of the conformed region. Particles in green are unconformed. Grasp point at the assigned location is highlighted by a magenta marker. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

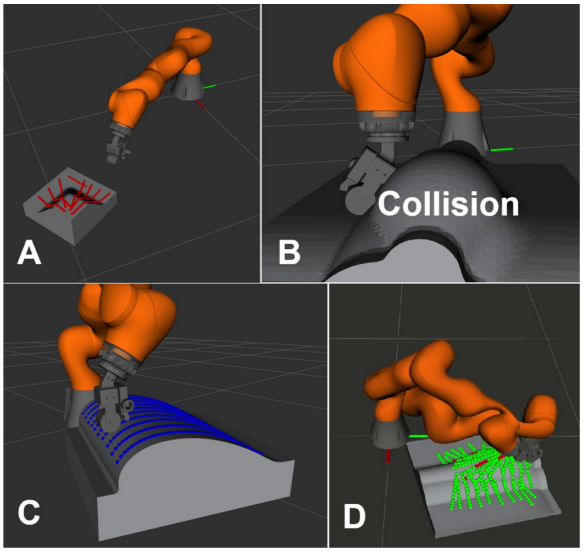

Fig. 20. (A) Waypoint are out of robot workspace. (B) Collision constraint is violated. (C) Velocity is not met. (D) Discontinuity exists in a path.

of waypoints $\{w_1^k, w_2^k, ..., w_i^k\}$. The number of waypoints i can be different for the paths. The path can belong to either of $\mathcal{S}_D$ or $\mathcal{S}_G$. Each waypoint represents the pose (position and orientation) in which the robot end-effector will need to align itself. Waypoints are defined in a local frame attached to the mold.

A manipulator (i.e., a robot) is fully defined by identifying its joint configuration in the configuration space. For an n-DOF manipulator, n joint angles will represent the joint configuration vector $\Theta = \{\theta_1, \theta_2, \cdots, \theta_n\}$. Joint velocity and the joint torques are represented as $\dot{\Theta}$ and τ respectively. We attach a tool center point (TCP) frame to the tool or end-effector of the robot. Robot aligns this TCP with the waypoint within some tolerances while executing the tool paths. We will be using the words workpiece and the mold interchangeably in this work.

A robot needs to be placed with respect to the workpiece such that it is reachable in position and orientation. Process constraints also need to be met. Solving the problem becomes challenging, as the workspace of a robot is limited. The constraints also have complex interactions between them. For instance, the regions where a robot can apply large amounts of force may be limited by the magnitude of velocity it can sustain.

The robot must also meet the process constraints and avoid collision and singularities. Another example is the continuity constraint, which forces the robot to execute the paths without large joint angle changes. Fig. 20(D) shows a workpiece on which continuity is violated by other constraints that are satisfied. The constraints velocity, force, continuity, collision, and singularity, are a function of the robot joint configuration.

A placement is feasible when joint configurations can be generated for all the waypoints. We need to solve an expensive inverse kinematics (IK) problem to find these joint configurations, which can minimize the overall reachability error. We can perturb the workpiece pose in the workspace of the robot until successful IK solutions can be obtained. We call the solution thus obtained as an *exact solution*. In order to find this exact solution, a non-linear optimization problem calls the IK function for cost evaluation. However, this problem is computationally intractable due to the following reasons. Any optimization algorithm is susceptible to local minima. All constraints have variations in the workspace of the robot. Depending upon the initial placement, the gradient can strongly lead to regions where one or few constraints are satisfied. The region where a solution exists can also be far from the initial guess, and it may take several iterations before it can be found. A large number of initial samples will, therefore, be required to be tested before one closer to a feasible configuration space region is generated. Such a sample will then lead to the solution. As for the size of the paths and DOF of the robot increase, the computational complexity of this problem gets intensified.

Fig. 20 illustrates this concept using bad workpiece placements. In the figure, A shows a part that is not reachable in position, so a large number of iterations are required to glide it into the workspace before a local minimum is found near the boundary where constraints are not satisfied. In B, C, and D, the part is reachable in position and orientation, and hence, feasible joint configurations exist. But the collision, velocity, and continuity constraints are not upheld, respectively.

The problem of workpiece placement has a 6-dimensional Cartesian state space. But, manufacturing processes like the composite ply layup involve the use of a large and heavy mold. Often due to physical constraints or to avoid using special fixtures, the mold is placed on the floor or table, and robots can be placed around it. This limits the DOF of the mold in the workspace to only 3 dimensions $< x, y, q0, 0, 0, q3 >$. $< x, y >$ is the translation along the horizontal plane (floor or a table) and $< q0, 0, 0, q3 >$ describes the rotation of the mold about a vertical axis Z of the global reference frame. The state space in an IK problem to evaluate the reachability of the robot will be dependent on the DOF of the robot. For our case, we use a KUKA iiwa robot having 7DOF. So the configuration space has 7 dimensions to it. Our approach, however, can be used for robots having more than 7DOF as well.

7.2. Problem formulation

We need to identify the homogeneous transformations of all the robots with respect to a reference frame. We take the world frame as a reference frame for the cell. The world frame coincides with the mold reference frame. The spatial transformations are written as ${}^{W}T_M$, ${}^{W}T_D$, ${}^{W}T_{G1}$, and ${}^{W}T_{G2}$ for the mold, draping robot, grasping robot-1, and grasping robot-2 respectively. ${}^{W}T_M$ is identity. D, $G1$, $G2$ are the robot base frames. We can then represent the mold in any robot frame using the equation ${}^{R}T_M = {}^{W}T_D^{-1} * {}^{W}T_M$, where R is the base frame of a robot. This will describe the pose of all waypoints in the robot base frame. Robot trajectories can then be generated by solving the inverse kinematics (IK) problem and finding the configuration Θ_i for an ith waypoint. All constraints, including the process constraints, are enforced while solving the IK and a feasible trajectory that satisfies the constraints will exist when joint configurations can be found for every waypoint in the tool paths. We are interested in finding the spatial transformations ${}^{W}T_M$, ${}^{W}T_D$, ${}^{W}T_G1$, ${}^{W}T_G2$, given the robot description (*robot*), CAD model of the mold, and the process requirements. We formulate this problem as follows:

$$
\begin{aligned}
&find\ {}^{W}T_b, \forall\ \ b = \{M, D, G1, G2\} \\
&\quad s.\,t.\ \ lb_i^k \le\ g_i^k(\Theta) \le ub_i^k \\
&\qquad\qquad h_i^k(\Theta) \le 0 \\
&\quad \forall\ w_i^k \in S_k, \forall\ S_k \in \mathcal{S}_d, \mathcal{S}_g
\end{aligned}
$$

Here, $g_i^k(\Theta)$ and $h_i^k(\Theta)$ represent the inequality constraints for the ith joint configuration in the trajectory corresponding to the path S_k. The motion of the mold is restricted to a feasible set of $\mathcal{X}_f$. $\mathcal{X}_f$ comprises of a feasible position in XY plane and orientation about Z that the mold can take. If necessary height adjustments are available, an additional translation along the Z-axis can be permitted.

7.3. Capability map

Explicit generation of capability maps can significantly improve the computation time for the robot placement (see Section 7.4 for details). This section will discuss the approach for generating capability maps.

Capability map C captures the position and orientation information of a manipulator throughout its workspace. This map can be used as an approximation to the positional and orientational reachability instead

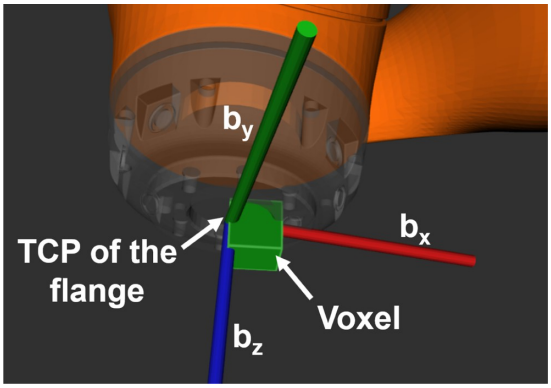

Fig. 21. TCP attached to the flange represented by three unit direction vectorsalong X, Y, and Z axes. Nearest voxel to the position of TCP is illustrated by a cube.

of computing an exact IK solution. The robot positional workspace boundaries can also be explicitly defined using this map. The position is encoded in a discrete approximation of the workspace. In order to generate the map, one can solve an expensive IK problem over uniformly sampled orientations for every discrete position in the workspace and mark the reachability in the underlying data structure. However, computing a high fidelity map can be expensive, and therefore, we randomly sample joint configurations to obtain reachable orientations of the robot.

1. *Sampling in Configuration Space:* A randomly sampled joint configuration gives the position and orientation of the robot flange with respect to the robot base in the workspace. The position can be mapped to the nearest discrete voxel in the space. Fig. 21 illustrates the TCP attached to the robot flange and its nearest voxel as a green cube. TCP orientations are represented with the unit direction vectors $< bx, by, bz >$ along the X, Y, and Z axes of the flange frame. The orientations can be appended to the list of previously sampled orientations in the voxel. We also maintain a list of manipulability indices for all sampled configurations in the corresponding voxel.
2. *Ensuring uniformity:* As the position and orientations are randomly sampled; we need to examine the neighborhood of all the voxels in the workspace. If the neighboring voxels of the voxel in consideration have a higher number of samples, then we improve the density in that voxel by solving IK over sampled orientations. These orientations are uniformly sampled on a sphere with the position of the sparsely populated voxel as the center.
3. *Sampling near workspace boundaries:* To ensure that a voxel is unreachable at the workspace boundary, we solve IK over uniformly generated orientations in the boundary voxels and guarantee that the voxel is truly unreachable. This distinctly identifies the workspace boundary for the robot.
4. *Generalizing orientations to axis-angle:* All the unit vectors along the X, Y, and Z direction of the flange frame for the reachable orientations are captured for each voxel. A set of unit vectors along a unique direction X or Y or Z can be represented by using a principal axis and two angles β_1, β_2. Fig. 22 shows the cross-section of a unit sphere and few of the sampled orientations along the Z-axis that are sampled in a voxel. β_1 and β_2 mark the inclusive region relative to a principal axis, which encloses all the orientations reachable in that voxel. $\beta_1, \beta_2 \in [0, \pi]$ radians. Thus when the queried orientation lies in between these angles, it is marked reachable by the robot. In terms of storage, we need to store the principal axis and two-radian angles from the actual map. Thus the C used occupies very low memory. The map we generated for KUKA iiwa 7 robot occupied 5398 KB of memory.
5. *Filtering singularities:* We can compute the average manipulability index of the robot for every voxel. Fig. 23 shows the variation of this index throughout the workspace of the robot. We can explicitly filter the voxels having a low manipulability index and omit them from C. This can avoid placing the part in regions where the robot will hit the singularity.

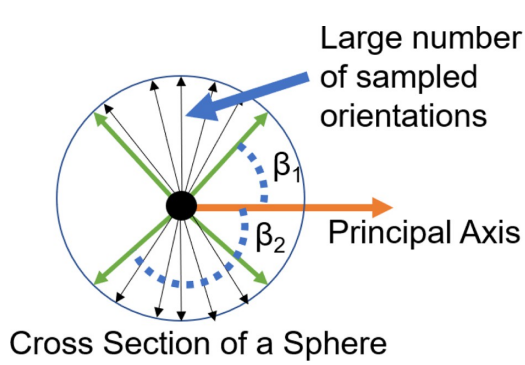

Fig. 22. Cross section of a sphere showing a few sampled orientations along the Z axis of the flange TCP in a voxel. A principal axis and two angles can represent these vectors collectively.

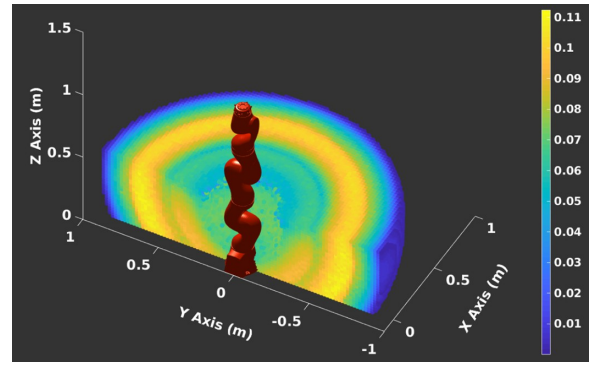

Fig. 23. Variation of the manipulability index throughout the robot workspace is shown. A higher manipulability index is desirable to avoid singularities.

C is computed for the robot flange. It is not required to compute the map again when a tool is attached. We can use a rigid body transformation to obtain the pose of the flange given the pose of the tool. As the map C needs to be pre-computed only once for a robot, computational performance of the algorithm is not affected. We also present some results which show the benefits of using C by benchmarking the computation time required to access the map compared to solving an IK problem. One thousand folds improvement in efficiency is obtained. Table 4 shows two types of comparisons. We first check if a randomly sampled feasible configuration is reachable in C. We tested against 1 million samples, and the map was successfully able to verify all solutions. This confirmed that no information was lost while representing the sampled orientations by an axis and two angles β_1, β_2. Next, a random orientation is sampled and verified if an IK solution existed. For 50,000 sampled orientations, we observed about 3% overprediction by the map. This map is taken as an input in our approach. We will now discuss an overview of our approach with a block diagram.

7.4. Overview of approach

Fig. 24 shows the block diagram for our approach. We have a two-layered approach where the first layer quickly explores the workspace of the robot for generating promising samples. Promising samples are then passed to the second layer, which finds an exact IK solution to the problem under constraints (Generate exact solution in the block diagram). The generation of promising samples uses the capability map to

Table 4
Table illustrates if a feasible orientation can be verified by the capability map. 1 million joint configurations are randomly sampled. All are verified by the map. Results on comparing the prediction of the map with an IK solver are provided. 50,000 random orientations are used for this test.

Capability Map Prediction Accuracy					
No. of Reachable Samples	% Reachable by Map	No. of Random Samples	% Reachable by Map	% Reachable by IK	% Over-prediction
1 Million	100	50,000	33.42	32.4	3.05

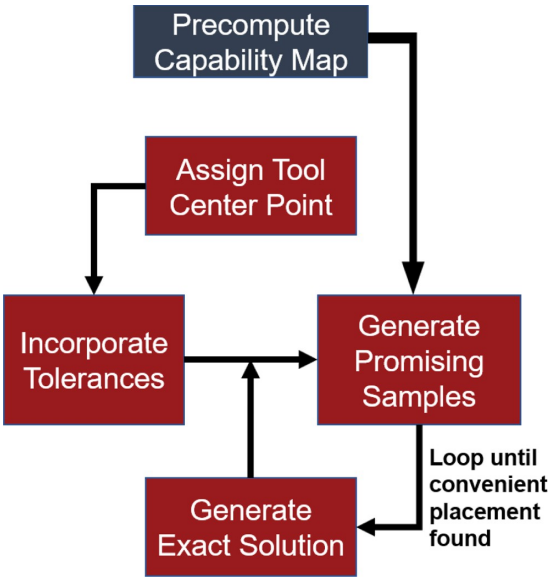

Fig. 24. Architecture of the robot placement algorithm illustrating the different blocks used.

approximate the solutions without solving IK. This layer can quickly prune regions of the workspace where solutions may not exist. The promising samples are passed to the second layer, where IK is solved to generate an exact solution. If a solution does not exist, then the next promising sample is chosen. Discussion of layer-1 or generation of promising samples is done in Section 7.7.1 and layer-2 is discussed in Section 7.7.2.

In addition, we also exploit the semi-constrained motion of the tool TCP over the mold. The DOF required by the process is less than the DOF of the robot. In our case, the process demands the only 5DOF, and the robot has 7DOF. In such cases, the tool paths are semi-constrained, and the chances of finding an IK solution increase. Tolerances at the tool TCP allow us to take advantage of semi-constrained tool paths. The concept of tolerances is discussed in detail in Section 7.5. We incorporate these tolerances and generate the new waypoints and provide them as an input to the first layer to generate promising samples (see Fig. 24). The tool geometry is also exploited to assign different tool center points (TCPs) for each waypoint (Assign tool center point in the block diagram in Fig. 24). If we use a different TCP for waypoints, the reorientation of the flange is reduced, which improves the probability of finding solutions. The tool TCP assignment is discussed in detail in Section 7.6.

7.5. Tolerances about the TCP

We define the concept of 1-axis and 2-axis tolerances. Fig. 25 illustrates these tolerances. For 1-axis tolerance, we use the example of a roller tool used for the composite layup process. A TCP is shown attached to the roller at its bottom. The roller must align the unit direction vector Y (green) axis of the TCP with the corresponding axes of the waypoint. But due to the cylindrical geometry, it can align the Z-axis (blue) within some radian angle tolerance. This generates a 2D tolerance cone at the TCP. Similarly, some tools can have tolerances about

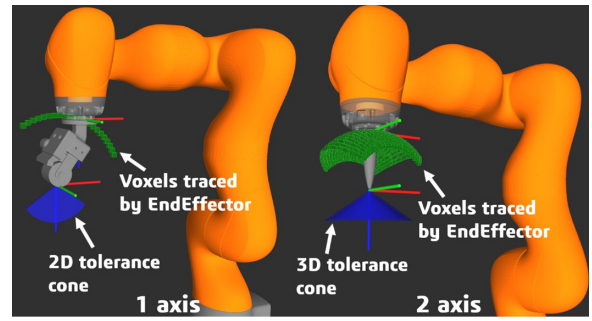

Fig. 25. 1 and 2 axis tolerances are shown. The Z axis of the TCP of the roller tool traces a 2D cone and Z axis of the TCP of the probing tool traces a 3D cone. Voxels traced by the flange corresponding to the orientations in the 2D and 3D cones are shown [61].

two unit direction vectors, as well. 2 axis illustration in the figure shows an example of a probing tool where the objective is to make contact at the tip of the tool. The orientation of the tool TCP does not really matter in this case. TCP of the tool generates a 3D tolerance cone.

We need to incorporate the tolerances with the approximate and exact layers of finding placement solutions. We discuss the method for incorporating tolerances in IK in detail in Section 7.7.2. We will first discuss how we can query the capability map and use it with tolerances to generate promising samples. In the case of tolerances, a set of discrete orientations can be defined within the 2D and 3D tolerance cones. As the robot reorients the end-effector within the tolerance cones, the flange traverses the corresponding set of voxels in the capability map. The flange traverses a 2D arc and a 3D surface while orienting the end-effector within the 1 and 2 axis tolerance cones. Fig. 25 illustrates this concept. A 2D arc is traversed when using the roller for composite layup application. This arc will correspond to a set of voxels in the capability map. All these voxels are then queried for the corresponding discrete orientation vector. As the same waypoint can now be approached from multiple orientations, feasible configuration space increases. As a result of this, the probability of finding solutions improves.

7.6. Assign tool center point

For simplicity, we consider that the roller will make line contact with the mold. A TCP is assigned on its surface, which is aligned with a waypoint on the mold. This TCP will lie at the center along the width of the roller. We can generate more TCPs along a circle formed by the intersecting a plane passing through the center of the cylinder and the cylindrical surface. This circle can be seen on the roller tool surface in Fig. 26. Finally, we discretize this circle into points located 10° apart from each other. These discretized points, along with their respective coordinate frames, are seen in Fig. 26. The z-axis of the coordinate frames at each of the discretized points is pointing in the outward radial direction, and the x-axis is in the tangential direction.

It is common to use different TCPs for different tasks. In this work, we developed an approach to automatically assign different TCPs along a path so that the robot can be reachable throughout the path. We vary the TCP that is used for a waypoint while executing a path. Consider the analogy to a human operator holding a tool and performing a task. The operator minimize the orientation change about the wrist, and

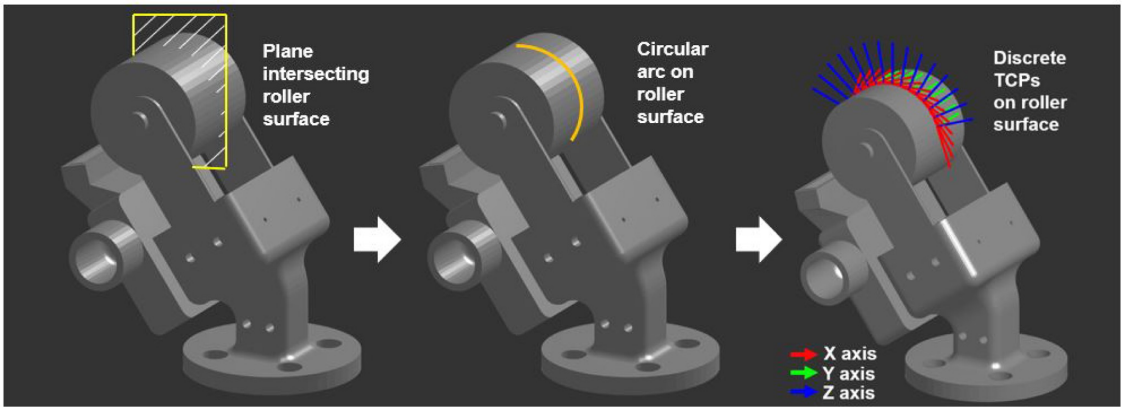

Fig. 26. The three step process showcasing the sampling of the tool surface to generate TCP candidates for the TCP assignment/selection algorithm.

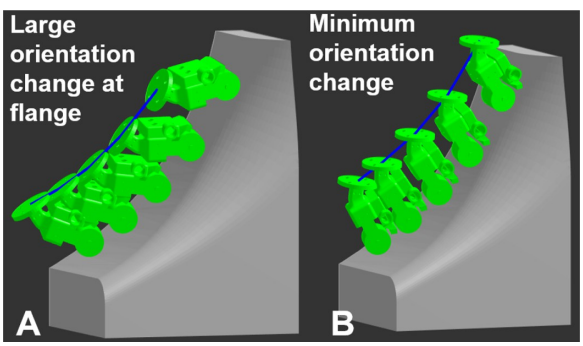

Fig. 27. A) The tool flange orientation change with out the TCP selection algorithm B) the tool flange orientation change (minimal) with the TCP selection algorithm.

unknowingly change the TCPs that come in contact with the workpiece. If we can minimize the orientation changes at the flange TCP, the robot reachability will increase. Fig. 27 shows two different paths executed by the flange. A shows the path which uses a single TCP throughout the path, and we can see that the flange has drastic orientation changes. B shows the case where the flange performs a smoother motion. The probability of the robot meeting constraints in such a case is higher. We need to find a TCP for each waypoint such that any collision between the tool and the mold is also avoided. We solve the problem given by the Eq. (6) in order to find the right sequence of TCPs along the path. More details about our work in this area can be found in [8].

$$\vec{sq}_{opt} \leftarrow \arg\min_{\vec{sq}}(OriCost(\vec{sq}, tool, \mathcal{M}_v)) \quad s.\ t.\ CollisionCost = 0 \tag{6}$$

Where $\vec{sq}$ is the sequence of TCPs, OriCost is the orientation change cost of the flange, the tool is a spherical approximation of the tool used for collision checks, and $\mathcal{M}_v$ is the EDT map of the mold. We solve this problem by using a state space search. We will now describe the steps in the algorithm used.

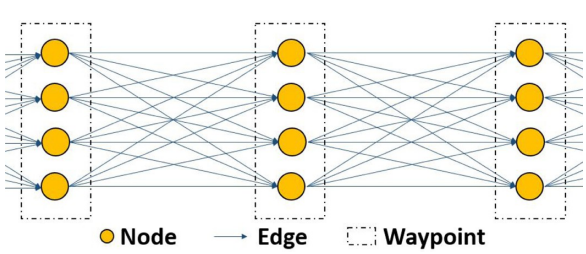

Fig. 28. Illustration of the acyclic directed search graph used for TCP selection ([8]).

1. *Node selection and edge creation:* A directed acyclic graph is formed of nodes and children nodes. The edges represent the relationship between the parent and child nodes. In our case, each sampled TCP of the tool at a position waypoint is represented using a node. The edges connect each parent node at a position waypoint with all the children nodes at the next consecutive waypoint, and they are directed in the same direction (see Fig. 28). Here, the total number of nodes in the graph is equal to *the number of nodes at each waypoint* $\times$ *the number of waypoints*. And the number of directed edges connecting the nodes in the graph is equal to *(the number of nodes at each waypoint)*^{2} $\times$ *(the number of waypoints-1)*.
   For a point-contact tool, different orientations of the tool represent the nodes at a waypoint. But for the roller tool along with different orientations of the tool, we can change the location of the TCP origin over the tool contact line (see the orange arc in Fig. 26). This allows us to have the ability to perturb the TCP position along with reorienting them. It makes the graph generation computationally expensive, but it allows us to have better setup placement using the best TCP for a waypoint.
2. *Cost Representation:* Each of the edges has an associated cost for traveling from the parent node to the child node of the edge. In our case, if a collision is occurring between the tool and the mold at the child node, we assign the edge at a very high cost. If there is no collision, the cost is equal to the angle between the unit Z vector *bz* of the flange TCP, and a vertical Z-axis is taken as a reference. This allows us to have the robot end-effector always facing down.
   The collision detection between the mold and the tool is determined using the EDT map. In our case, we approximate the tool surface using a set of spheres (see Fig. 29). The mold is represented by voxelising the environment $\mathcal{M}_v$. The tool collides with the mold if the closest voxel on the mold is intersecting with a sphere representing the tool.
3. *Initialization and shortest path determination:* The shortest path algorithm (e.g., Dijkstra) is initialized with each node of the first

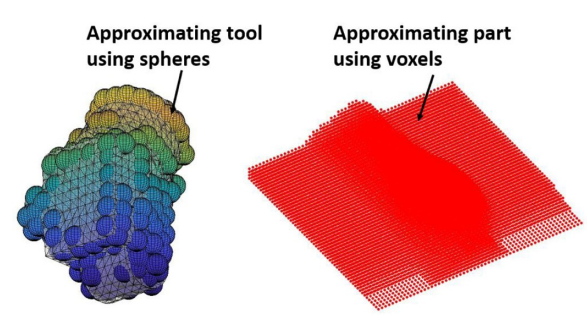

Fig. 29. Illustration of the of tool body approximation using spheres and the mold body approximation using voxels [8].

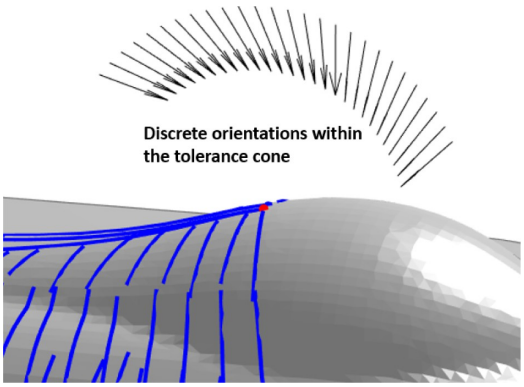

Fig. 30. Discrete orientations for the flange are sampled for a tolerance cone corresponding to a waypoint on a mold. The roller TCP is rotated at discrete angles defined within tolerances to get these orientations.

position waypoint, and it is terminated once it reaches any node on the final position waypoint. The path with the lowest cost among all these shortest paths is the global minimum for the designed graph. Moreover, if the path exists, the algorithm will guarantee the lowest cost path with no collision.

7.7. Solution strategy based on successive application of constraints

7.7.1. Layer-1: Generate promising samples

The constraints which are considered in this layer are position, orientation, a tool to mold collision, and singularity. Fig. 30 shows the discrete Z orientations of a flange for a waypoint. These orientations are obtained by traversing the TCP using 1 axis tolerance. Each orientation originates from the center of a voxel. The robot flange needs to position itself at the center of a voxel. The Z-axis of the flange should align with an orientation vector. When the discrete orientations are generated within the tolerance cones, the collision between the end-effector and the mold is also checked.

We randomly generate a large number of samples in $\mathcal{X}_f$. For every sample, the waypoints on all the paths are transformed in the robot workspace. We can then check if all the waypoints are within the boundaries of the robot. Appropriate transformation can be applied to the position and orientation of the discrete Z orientations in the tolerance cones. The orientations for which the tool collides with the mold are marked in the underlying data structure and will never be used to query the capability map. Fig. 31 shows a mold with the tool at three orientations within the tolerance cone. Orientations in which tool collides are marked in red. The tolerance cone size is adjusted based on this collision data, and the tool never collides with the mold until the Z-axis of tool TCP is within the tolerance cone. Exact IK evaluation will include the collision of the robot with the environment.

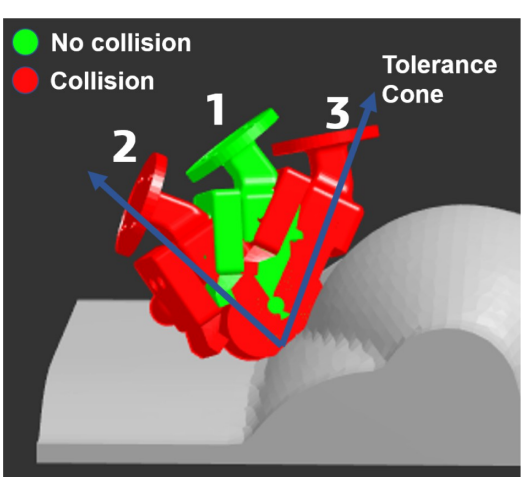

Fig. 31. Three orientations for the roller are shown within a tolerance cone. The orientations in which the roller can collide are discarded.

All the voxels in the map C are either reachable or unreachable. The map C captures the robot workspace boundaries. This will guarantee that the waypoints will never violate the position constraint. Even though the waypoints will be reachable in position, there will be violations in the orientation. For a tolerance cone, a voxel is queried for the corresponding orientation vector. If it is within the cone represented by an axis and two angles β_1, β_2, then the waypoint is marked to be reachable. The total number of waypoints violating orientation constraint are computed for a sample in this manner. The generation of these random samples is computationally inexpensive even though a large number of orientations are present due to available tolerances. In order to filter good placement samples, we provide the following two scores to each sample. The number of waypoint violations and an overall reachability score. Fig. 30 shows a number of orientations for the flange within a tolerance cone. The overall reachability for this tolerance cone will be the number of collision-free orientations reachable in the capability map. Overall reachability score over all the waypoints is the total number of orientations reachable in all the tolerances cones combined. The two scores are different as a waypoint is reachable even if one orientation in its tolerance cone is reachable. A sample can have higher overall reachability, but one waypoint is not reachable. So our objective is to have samples that have zero waypoint violations and high overall reachability.

High overall reachability implies that a number of orientations are feasible for the given waypoint, which will improve the probability of the existence of a solution. The randomly generated samples are then sorted based on these scores and selected for further refinement. A state space search is conducted in the feasible workspace $\mathcal{X}_f$ using a promising sample. If the number of waypoint violations is not zero, we can minimize these violations and maximize the overall reachability using a state space search. In our implementation, we used a simple gradient descent technique where feasible neighboring states of the sample are explored. If a potentially better sample is present, we modify the current sample. We iterate until a local minimum is reached.

In addition, we can also formulate a multi-objective function for maximizing a performance measure along with overall reachability. Kinematic and dynamic manipulability are two popular measures in use. These measures are stored while computing C. Manipulability over a waypoint can be estimated by considering the manipulability in the set of voxels made within the tolerance cone. The promising sample ($\vec{x}_{init}$, where $\vec{x}$ is a pose of the mold with respect to the corresponding robot base) generated by this approach, is then passed to layer-2 for checking if an exact solution (IK) exists.

During the computation of the map C, we also compute the Jacobian of the robot at the sampled configuration. The Jacobian not only provides a means to compute the performance indices, but the maximum achievable velocity vector can also be computed. MVR (manipulator velocity ratio) is based on this concept. A set of the magnitude and direction of such velocity vectors is stored for a given voxel. A similar approach is followed for storing the force-moment vector. This gives us an upper bound on the velocity, which can be achieved by the robot. Given a manufacturing process constraint of velocity and force, our algorithm can guarantee that a solution will not exist if the robot is not capable of meeting the requirements. The algorithm will also guarantee that all the waypoints are within the robot workspace boundary as the boundary is explicitly identified in the map C.

7.7.2. Layer-2: Generate exact solution

Finding an exact solution involves solving a non-linear optimization problem in the configuration space for every waypoint in a path. We propose a path-constrained trajectory planner that will evaluate reachability over the paths by enforcing the constraints. All the

waypoints are transformed with respect to the base of the robot. An optimization problem given by the Eq. (7) is solved to find the joint configuration Θ_i^k for the ith transformed waypoint in the kth path.

$$\Theta_i^k \leftarrow \arg\min_{\Theta}(\mathcal{PE}(\Theta, \Theta_{guess}, w_i^k, {}^R T_M, robot)) \tag{7}$$

$$\begin{aligned} s.\ t.\ \ & cosine^{-1}(bz^T * wz) - toleranceangle \leq 0 \\ & lb_i^k \leq g_i^k(\Theta) \leq ub_i^k \\ & h_i^k(\Theta) \leq 0 \\ & \forall\ w_i^k \in S_k \end{aligned} \tag{8}$$

Here, $\mathcal{PE}$ returns the error in the pose of the roller TCP attached to the robot and the waypoint. The TCP used for the waypoint in consideration is obtained from the TCP assignment routine. $\mathcal{PE}$ is found by using the expression $e'*W*e$, where e' is the transpose of vector e. Error vector e is a 4×1 vector constituting the following components. Three components of the error in position and one component of error in Y orientation. Let bx, by, bz be the unit direction vectors of the TCP and wx, wy, wz be the unit direction vectors of the waypoint on the surface of the mold. Thus error in the orientation of the Y-axis is given by $1 - by^T wy$, where by' is the transpose of vector by. We need to account for tolerances while computing the orientation error. As described earlier, the Y axes of the TCP of the roller must align exactly with the corresponding axes of the waypoint. There is some tolerance in the angle which the unit Z-direction vector of the roller TCP can make with the Z-axis of the waypoint. Eq. (8) takes care of the tolerance in Z-axis alignment, where *toleranceangle* is the radian angle between the two-unit Z-direction vectors.

Θ_{guess} is an initial guess passed to the algorithm. We pass the joint configuration of the previous waypoint in the path as an initial guess. $g_i^k(\Theta)$ and $h_i^k(\Theta)$ are the inequality constraints enforced. lb_i^k and ub_i^k are the corresponding lower and upper bounds. We will now discuss smooth constraint violation functions that we propose to compute the hessian and gradient for the non-linear optimization problem. Following is the list of constraints enforced and the corresponding functions used.

1. *Configuration Space Constraints:* The joint angles, joint velocities, and joint torques of the robot are limited within the upper and lower bounds. Equations $\Theta_{lb} \leq \Theta \leq \Theta_{ub}$, $\dot{\Theta}_{lb} \leq \dot{\Theta} \leq \dot{\Theta}_{ub}$, and $\tau_{lb} \leq \tau \leq \tau_{ub}$ enforce these inequalities.
2. *Instantaneous Velocity Constraint:* We need to ensure that the robot must meet the minimum desired velocity while executing the paths. For a waypoint w_i^k, we can apply a rigid body transformation to find the corresponding robot flange pose f_i^k. An instantaneous velocity vector v_i^k can be defined directed from f_i^k to f_{i+1}^k. v_i^k is comprised of position and orientation velocities. Θ_i^k should be such that the robot can achieve the instantaneous velocity required at the waypoint w_i^k. A Jacobian matrix $\mathcal{J}$ is used to express the differential kinematics of the robot. We can define the instantaneous velocity constraint at any configuration by the equation $\dot{\Theta}_{lb} \leq \mathcal{J}^{\dagger} v_i^k \leq \dot{\Theta}_{ub}$. $\mathcal{J}^{\dagger}$ is the Moore-Penrose pseudoinverse.
3. *Transition Velocity Constraint:* Instantaneous velocity constraint does not ensure if the joint velocity is sufficient enough to transition from Θ_{i-1}^k to Θ_i^k. We introduce a transition constraint for this purpose. Let t_i^k be the time taken by the robot to transition from w_{i-1}^k to w_i^k. As the velocity required by the process is provided, we can find the time taken for a small segment of the path traced in between these waypoints. Using this t_i^k, we can give the equation for transition constraint as:

$$\forall\ j \in n, (\theta_i^j - \theta_{i-1}^j)/\dot{\theta}^j \leq t_i^k \tag{9}$$

   Here, $\dot{\theta}^j$ is the magnitude of the maximum velocity of the jth joint in the configuration. n is the DOF of the robot.
4. *Continuity Constraint:* We need to avoid any large joint angle changes while executing the paths. The Jacobian matrix can be used for two close configurations are similar. We use pearson's correlation coefficient [16] between the Jacobian matrices at consecutive configurations Θ_{i-1}^k and Θ_i^k to enforce the continuity constraint given by the equation $-\ correlation(\mathcal{J}_{i-1}, \mathcal{J}_i) \leq \delta$.
5. *Force Constraint:* The robot must apply the desired force on the path which is being executed. The Jacobian matrix establishes the relationship between the joint torque vector τ and the force moment vector $\vec{F}$ at the end-effector. Assuming the inertial effects to be negligible under constant velocity, we use the Jacobian matrix to enforce the constraint on the joint configuration of the robot. The force constraint can be given by the equation $\tau_{lb} \leq \mathcal{J}^T \vec{F} \leq \tau_{ub}$.
6. *Collision Constraint:* We model the robot and the roller tool using a set of spheres. This approach is inspired by the work done in [84]. An EDT map is computed for the environment for faster collision detection. We then enforce the constraint given by the Eq. (10) over all the representative spheres.

$$\forall\ Spheres, clearance - (d - r) \leq 0 \tag{10}$$

   where *clearance* is the minimum clearance between the rigid bodies. d is the minimum distance of the corresponding sphere, and r is the radius.
7. *Singularity:* Manipulability index σ [95,96] is used to keep the robot away from singularities. We can analytically compute this index at any given configuration using the equation $\sqrt{det(J, J^T)}$. We use the equation $-\sigma + \delta \leq 0$ to enforce the singularity constraint at any configuration.

A solution, if exists, may not be found by the optimization algorithm solving the Eq. (7) if all the constraints are enforced at once. In our approach, we apply the position, orientation, and continuity constraint first to guide the end-effector in the region where position and orientation constraint is guaranteed to be satisfied. Continuity constraint is more relaxed as it directs the algorithm to find a solution in position and orientation regions, which is closer to the previous solution. Singularity constraint, on the other hand, will truncate the region to avoid singular solutions. So continuity and singularity constraints are applied along with position and orientation.

We then apply velocity and force constraints. The constraint which imposes the most restrictions is the collision constraint. Due to complex geometries, the tool is likely to collide with the mold. So we enforce the collision constraint in the end, which will have the smallest feasible region. We show how such an approach can be better in terms of success rate and computation time in the results section. However, even after using a successive application of constraints strategy, it is not guaranteed to find a solution if it exists. It is highly dependent on the initial guess used in the optimization algorithm. If the initial guess is closer to a region where all the constraints except one are being satisfied, the algorithm can get stuck in the local minima.

In the second layer, we solve a non-linear optimization problem given by the Eq. (11). $\mathcal{RE}$ is the aggregated pose error over all the waypoints for a given workpiece placement. $\vec{x}$ represents the pose of the mold with respect to the robot base in consideration. $\vec{x}_{init}$ is the promising sample generated by layer-1, which can be passed as an initial seed to the algorithm. However, the evaluation of $\mathcal{RE}$ is computationally expensive. The function $\mathcal{RE}$ essentially solves for IK problem we described earlier under constraints. Every constraint evaluation function will require evaluation of the forward kinematics, Jacobian, its inverse, and collision detection. These evaluations are expensive, which causes the overall time to be higher. In our previous work in [61], we show the computation time taken for a workpiece used in composite layup using this approach is high. Hence for this work, we propose a different routine for bringing the time taken to find a solution within reasonable values. The exact evaluation of reachability will be used only to see if a trajectory can be generated using $\vec{x}_{init}$. As soon as the

trajectory fails at a couple of points, we terminate the evaluation and use another sample generated by layer-1. In practicality, layer-1 runs in parallel generating promising samples, and they get evaluated by layer-2. We propose an anytime algorithm that will keep generating solutions until some time-bound or the user has found a placement that is convenient to be used. The only drawback of this approach compared to using a non-linear optimization to perturb the workpiece pose in the neighborhood is that a solution that would have existed can be missed. For instance, if the promising sample required a small fine-tuning in order to make the trajectory feasible, using a state space search or optimization routine can be useful. One can couple a state space search as well. Although generating promising samples is computationally inexpensive and for complex workpieces, our proposed approach works well.

$$^{R}\vec{x}_M \leftarrow \arg\min_{x}(\mathcal{RE}(\vec{x}, \mathcal{S}, \vec{x}_{init}, robot)) \tag{11}$$

7.8. Using the algorithm to place the robots

7.8.1. Draping robot placement

The tool path followed by the draping robot is more complex compared to either of the grasping robots. The tool paths are on the mold and can have higher curvatures involved. The collision between the roller and the mold is another challenge. Draping robots must also apply the necessary force while executing the paths at prescribed velocity. All the constraints described earlier will be applied when generating trajectories for the draping robot. We first place the draping robot in the cell. The world frame is coincident with the draping robot. We then find the transformation $^{W}T_M$ using our approach.

7.8.2. Grasping robot placement

The placement of the grasping robots is different from the draping robot placement. The grasping robots need to apply a force that has very small magnitude and can be neglected during trajectory generation. This force is primarily required to keep the ply under some tension in certain regions. The paths do not make contact with the mold and are geometrically simpler. Most of the paths that are required to reposition or regrasp the gripper are straight lines. Another major difference exists due to the fact that is grasping robots are always under impedance control. The trajectory of the grasping robot will be altered in the vicinity of the initial trajectories in order to take care of contingency. A feasible placement must enable the robot to reach the grasp locations computed by the grasp planner. In addition, we need to increase the probability of the grasping robot being able to execute modified trajectories.

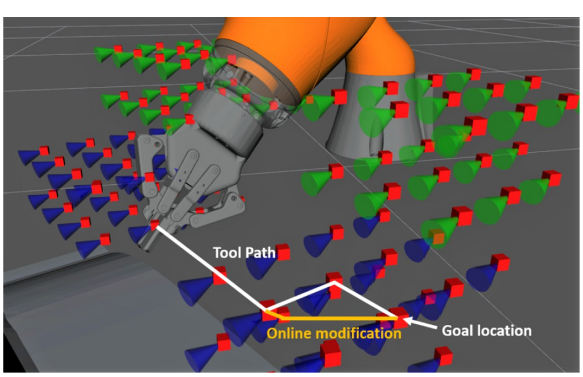

Fig. 32. Voxels in a neighborhood $\mathcal{N}$ of the grasp tool paths are shown as red cubes. Approach cones (blue) for the gripper TCP and the corresponding approach cones (green) for the flange are illustrated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Our algorithm discretizes the space around grasping locations using a three-dimensional array of cubes called voxels. We then define a neighborhood $\mathcal{N}$, which will encompass the regions where a modified trajectory could exist. $\mathcal{N}$ is essentially a set of voxels around the mold under consideration. Fig. 32 illustrates a mold with voxelized space around it. The voxels in $\mathcal{N}$ are shown in red. A repositioning or regrasping tool path can be mapped to the corresponding voxels. So for every grasping path, there is a discrete set of waypoints along the path. These waypoints are mapped to the centers of their corresponding voxels. Fig. 32 shows the computed tool path for the gripper mapped to the corresponding voxels illustrated with white lines. We can then use these waypoints and use our approach to find the suitable placement of the robot. The figure also shows a possible modification to this tool path, which can be required due to an online correction. We must ensure that the chances of generating feasible trajectories for such modifications are high.

Placements are found such that reachability is maximized in position and orientation over $\mathcal{N}$. We account for this overall reachability over $\mathcal{N}$ in layer-1 of the algorithm, where then generate an approximate solution. If all the voxels in $\mathcal{N}$ are within the workspace of the robot, $\mathcal{N}$ will be reachable in position. In order to estimate the reachability over orientation, a set of orientations is sampled in each voxel. These orientations are sampled within a 3D cone called an approach cone. Such approach cones are assigned to every voxel. An approach cone will have a mean unit direction vector and sample orientations around it within some angle. Fig. 32 illustrates this concept. The voxels in $\mathcal{N}$ are plotted in red, and their corresponding approach cones are plotted in blue. We use the map C to estimate if the position and orientation are reachable. So we need to transform all the orientations in the approach cones to the flange. Fig. 32 shows the voxels (red) and approach cones (green) where the flange frame needs to be reachable. Once a promising sample is found, having maximum reachability in these approach cones, it is passed to layer-2.

7.9. Computational results

Six molds were used to test the algorithm. Five molds are identical to the ones used to test the grasp planner. Fig. 33 shows the CAD models of these workpieces. A sixth mold F was introduced to test our algorithm against large workpieces having a vertical tool path. Mold D is also used for physical experiments. The mold was machined from a block created by adhering MDF (medium density fiberboard) plies together. The geometry of the mold is inspired by an industrial component of a medium scale geometry. As the overall mold is 820 × 570 × 75 mm, it is beyond the reachability of the current robots being used. Two patches of plies can be used to layup on the entire mold. The dimensions of molds A, B, C, D, and E are the same as mentioned in the computational results of the grasp planner. Dimensions of mold F are 300 × 350 × 500 mm. The Cartesian velocity and force constraints imposed by the process are 40 mm-s^{-1} and 40 N, respectively.

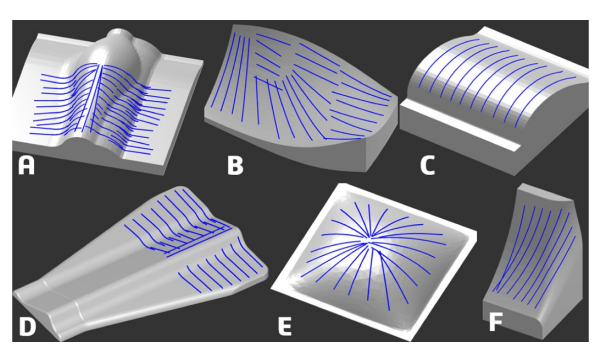

Fig. 33. Workpieces or molds used for testing the part placement algorithm.

Table 5
Average computation times (seconds) and success rate for baseline approaches based on 50 random initial samples used.

Average computation time (seconds) and number of successful solutions								
		Baseline Approach 1			Baseline Approach 2			
Mold	No. of Points	No. of successful samples	Avg. time to evaluate 1 sample	Time to generate 1 solution	No. of successful samples	Avg. time to evaluate 1 sample	Std. Dev	Time to generate 1 solution
A	524	1	2.45	122.43	20	1371.62	1329.71	3429.05
B	486	4	1.95	24.43	5	3237.86	1775.43	32378.60
C	470	1	1.42	71.14	40	364.67	738.13	455.83
D	479	3	2.72	45.38	18	1309.60	924.29	3637.78
E	357	7	1.86	13.29	40	62.62	97.70	78.28
F	413	3	0.93	15.43	15	1696.22	1983.71	5654.06

We are primarily interested in the computation time and success rate of finding a solution. The computations are performed using MATLAB and C++. Path-constrained trajectory was generated using C++ implementation. Generated placement is called as a sample while presenting the results. The algorithm is benchmarked against two baseline approaches. Baseline approach-1 generates some initial random samples in the workspace without using the precomputed capability map C. We then use our algorithm in layer-2 for exact evaluation of this sample. The sample is declared as a solution if a trajectory can be successfully generated. Baseline approach-2, on the other hand, uses the exact evaluation algorithm as an objective function to evaluate overall reachability across all the waypoints. A non-linear optimization problem given by the Eq. (11) is solved to find a feasible placement. We run the baseline and our approaches using 50 initial samples.

Table 5 summarizes the average computation time in seconds and the success rate for finding a solution for 50 random initial samples. In the case of baseline approach-1, we terminate any further evaluation of inverse kinematics if solutions are not obtained for at most two waypoints for a sample. The time per one sample is an average over all the 50 samples. However, we do not use a successive application of constraints strategy for solving IK. All the constraints are enforced at once. We have also computed the average time taken to generate one feasible solution on the basis of the success rate and the time taken per one sample.

Similar results are provided for baseline approach-2. However, as we are solving a non-linear optimization problem and for the objective function to be continuous, we cannot terminate IK if solutions are not obtained over at most two points. This drastically increases the computation time as a single call for exact evaluation reachability over all waypoints is expensive. The algorithm makes several such calls to evaluate the gradient and hessian in order to generate iterates. The high computation time for generating a single solution is thus justified. If a solution exists in the vicinity of the random sample, the time taken is substantially lower compared to the average computation time per sample. It is because of the workpiece being at the boundaries of the workspace; several iterations are required until the optimization algorithm converges to a local minimum. In most of the cases, a solution is not present in this local minima. This explains the high standard deviation in the data presented in Table 5 for baseline approach-2.

We can generate a promising sample using our approach in layer-1. In our previous work in [61], we used the promising sample and solved the non-linear optimization problem. Significant reduction in the computation time and improvement in success rate was observed as the non-linear optimization algorithm started with promising initial guess within the workspace and away from the singularity. The molds used in this work are bigger in size and more complex. As they have to be restricted on a horizontal surface, the feasible space reduces. Table 5 also shows that one is better off generating random samples until a solution is found. As a result, our approach is based on testing a promising sample using the exact evaluation function. If a trajectory exists, it is declared as a solution; otherwise, a new promising sample is picked for evaluation. The results for our approach are summarized in Table 6. These results are presented for the placement of a draping robot. Layer-1 comprises a generation of a random sample using C and then maximizing overall reachability within tolerance cones and performance measure, if any. The average computation time for a sample is presented for both stages. A successive application of constraints strategy is used to enforce constraints while computing the inverse kinematics. Even in this case, we terminate further evaluation of IK if solutions fail intermittently. We can observe that the number of successful samples out of 50 promising samples are higher for our approach. Despite, more success rate, the computation time for layer-2 or exact evaluation is lower. This is due to the fact that the constraints are applied successively instead of being enforced at once. The table also shows the improvement in computation time obtained against baseline approach 1 and 2.

Table 7 shows the benefits of using a successive application of constraints strategy while solving IK. We have evaluated the average computation time and success rate using three sequences of applying constraints for 50 different placements. All constraints will be enforced in constraint sequence-1. Constraint sequence-2 enforces all constraints except collision. In constraint sequence-3, position, orientation, singularity, and continuity are enforced first. The obtained joint configuration is used as an initial guess, and IK is re-solved with velocity and force constraints. If a collision occurs, the updated joint configuration is used as an initial guess, and the IK is resolved with collision enforced. IK is solved for all the waypoints in a path while computing these results. Table 7 shows that constraint sequence-2 provides significant improvement in the success rate for finding solutions and some improvement in computation time per sample. Constraint sequence-3, however, gives a significant improvement in computation time as well as success rate compared to all constraints enforced. So we use constraint sequence-3 for our computations.

Placement of the grasping robots involves the need for solving IK as only reachability must be ensured during the process. We use the capability map to find promising samples where maximum approach cones in the discrete voxels are reachable. We then use Ik to verify is the grasp locations, and tool paths generated by the grasp planner are reachable to find feasible placements. Table 8 shows these computation times for different layers.

8. Experimental results

8.1. Physical experiment

We conducted physical experiments using mold shown in Fig. 34. The figure also shows the draped region and virtual partition on the mold used in our experiments.

Fig. 35 shows the different simulation stages of grasp planning. Vertical motion of the mold with respect to the robots is allowed while computing placement, which offers more reachability. Thus, the mold is elevated in the physical setup. We first conveniently place the mold on the table, and the robot placements are determined. The physical setup

Table 6
Average computation times (seconds) and the success rate for our approach. Percentage improvement in computation time compared to baseline approaches are also presented.

Average computation time (seconds) and number of successful solutions for our approach								
		Layer-1 (times)		Layer-2 (time)			% Improvement in time	
Mold	No. of Points	Generate 1 promising sample	Maximize overall reachability	Exact evaluation	No of Successful Samples	Time to generate 1 solution	Compared to Baseline-1	Compared to Baseline-2
A	524	0.58	1.81	1.41	11	17.26	85.90	99.50
B	486	0.06	1.72	2.29	31	6.57	73.12	99.98
C	470	2.24	2.51	4.20	32	14.00	80.33	96.93
D	479	0.05	1.08	4.11	38	6.90	84.79	99.81
E	357	0.69	0.59	1.81	50	3.09	76.75	96.05
F	413	0.31	0.85	0.59	35	2.51	83.76	99.96

Table 7
Average computation time and success rate for different sequences of applying constraints while solving inverse kinematics.

	Constraint Sequence-1 (pvfc)		Constraint Sequence-2 (pvf+c)		Constraint Sequence-3 (p+vf+c)	
Mold	Number of successful samples	Avg. computation time for 1 sample	Number of successful samples	Avg. computation time for 1 sample	Number of successful samples	Avg. computation time for 1 sample
A	0	36.39	8	11.59	8	5.57
B	14	28.63	31	9.79	31	6.71
C	0	17.56	32	8.03	32	6.43
D	2	20.61	38	7.75	38	4.89
E	19	10.77	28	4.20	29	2.26
F	10	25.86	35	30.01	35	12.40

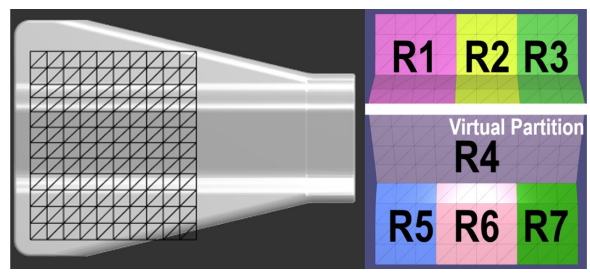

Fig. 34. Drape simulated over the mold used for physical experiments. Virtual partitions and the sequence of regions for this mold is also illustrated.

is shown in Fig. 2. We set the temperature of the air to 45° C, force applied to 35 N, end-effector stiffness along the X, Y, and Z axes to 3000,3000,2000 N-m^{-1} respectively, and the velocity of the robot to 15 mm-s^{-1}. Fig. 35 also shows the stages during the layup process. The grasping robots perform regrasps and reposition the end-effector according to the plans generated by our algorithm. We assume that some form of automation will exist to feed the pre-cut plies to the cell. In our experiments, an operator aligns the sheet with mold edges and feeds the edges of the sheet to the grippers.

There is some uncertainty between the simulated and the actual ply state. If the grasping robots do not react to the excess tension in the ply, the layup process may have defects. The grasping robots are therefore set in a compliance mode. By that, we mean that the robots are under impedance control mode, and if tension builds during execution, the robots comply and reposition the end-effector along the direction of the tension. This can prevent defects from forming during execution. Fig. 36 illustrates the difference between a layup performed under compliance and without compliance. We observe a high-cost ply state when robots do not comply and defects are present in the layup. On the other hand, compliance mode leads to reduced defects.

Even under compliance mode, there are small defects like a wrinkle that appears. The system analyzes this defect and calls for human intervention. The operation is halted and a human repairs the defect after which the operation can be continued. The grasping robots maintain their locations under compliance mode. The operator can adjust these locations if required while intervening. The grasping robots resume from new locations. Fig. 35 illustrates such a defect that was generated and was repaired by the human operator before proceeding with the next layer. No visual defects were present in the completed layup after human intervention. Completed layup using 15 plies is shown in Fig. 37. Table 9 also lists the time taken by each operation during the process. Some of the operations are to be done only once for a mold. Human intervention time includes the time taken by the operator to place the sheet and average time taken to repair defects throughout the process. We can see that the overall human input required for the process can be reduced by a significant amount for medium scale production. For instance, manufacturing 100 parts will result in a 88% reduction in human input. These results show that the process is

Table 8
Average computation time for grasping robot placements.

Average computation time for grasping robot placements						
	Grasping Robot - 1			Grasping Robot - 2		
Mold	Generation of promising sample	Maximize overall reachability	Exact evaluation	Generation of promising sample	Maximize overall reachability	Exact evaluation
A	0.04	0.02	9.60	0.04	0.04	8.26
C	0.03	0.01	8.15	0.04	0.01	7.88
D	0.05	0.02	11.88	0.04	0.02	11.39

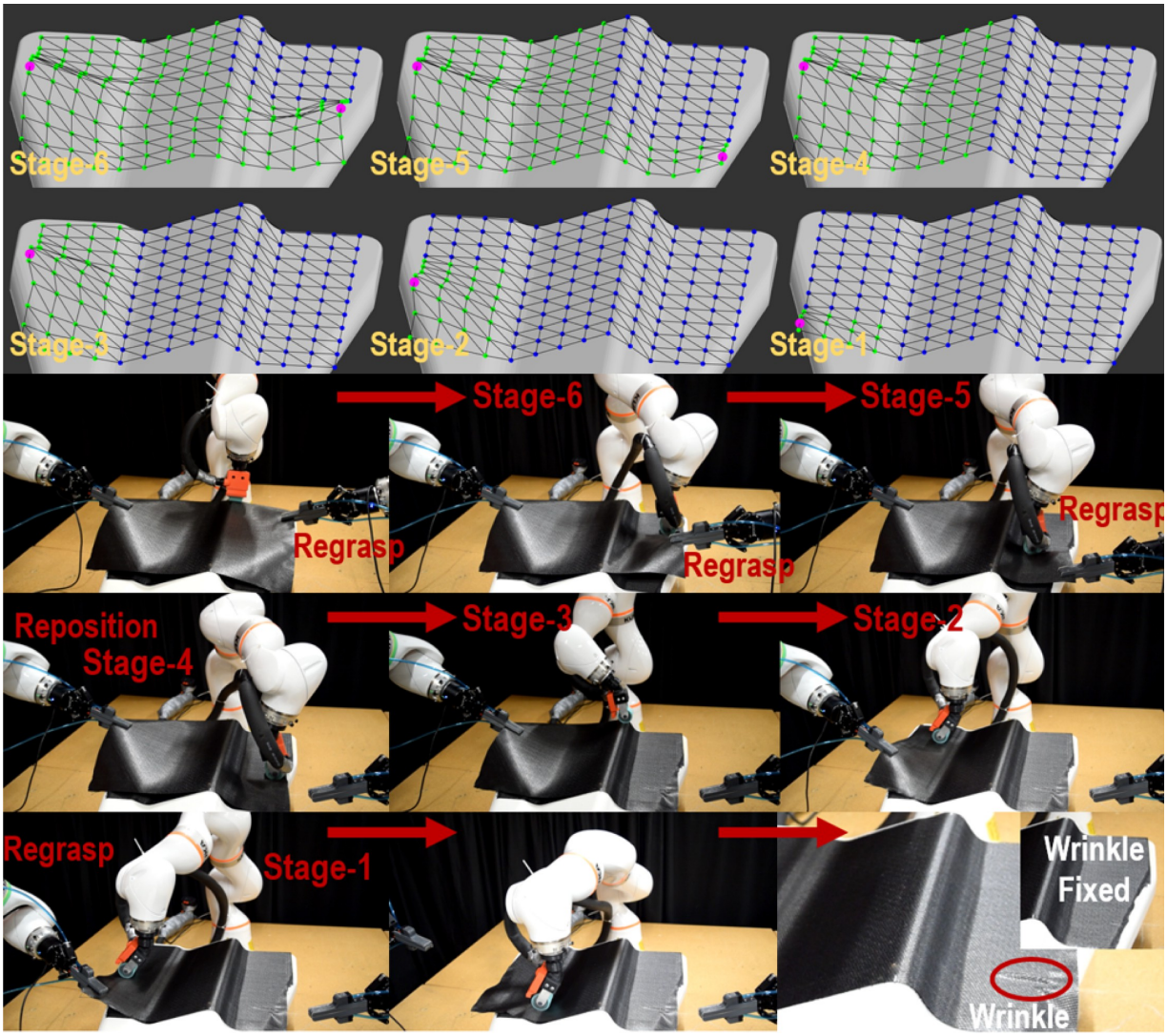

Fig. 35. Simulated ply states in the different stages of backward tracking search are illustrated. Instances of the layup process, the repositioning and regrasping motions performed by the grasping robots are also shown in the figure. Human interference is required to fix the wrinkle generated.

feasible to be automated and used for commercial purposes. The laminate was also inspected for defects. Resin distribution studies were done to tune the parameters of the process in our previous work in [59,62]. The work also demonstrates physical experiments on other smaller molds. In this work, we perform a visual inspection for air pockets and defects and primarily present the conformity and fiber alignment results from the laser scanner and apodius system, respectively (Fig. 38).

8.2. Conformity analysis

In order to ensure that the ply is conformed well to the mold and no air pockets are present, we conducted a conformity analysis on the mold. A point cloud is generated using hexagon absolute arm integrated with a laser scanner. This point cloud is compared to the reference point cloud, which represents the surface of the mold offset with the thickness of the laminate, respectively (Fig. 38). The deviation is within tolerance. The laminate with 15 sheets was scanned and maximum deviation was found to be 0.6 mm. The tight concave regions are usually difficult to layup as air does not get completely pushed out. The curing process applies vacuum pressure on the lamina, which removes the voids present.

8.3. Fiber alignment

We inspect the fiber alignment in the ply using the Apodius 2D sensor. The sensor is attached to the robot flange, and trajectories are generated so that the robot can scan the different regions of interest on the mold. Fig. 39 shows the draping robot with apodius sensor. The sensor analyzes the image and computes the deviation of fibers. We input a reference orientation to the software, and all the fiber angles are compared against this reference. A reference orientation is specified by the design engineer. In our case, our goal was to orient the fibers parallel to the edges of the mold. The distribution of fibers with respect to this reference is obtained using the software. Respective alignments for different points on the mold are collected and analyzed for defects. The fibers did not deviate at the concave regions of the mold during layup. Fiber angles were found to be within 0.1° angle. The process parameters were appropriate and did not cause any damage to the fibers in the right concave regions.

8.4. Visual inspection

The automated layup quality was comparable with hand layup. Air pockets or bridging were not observed.

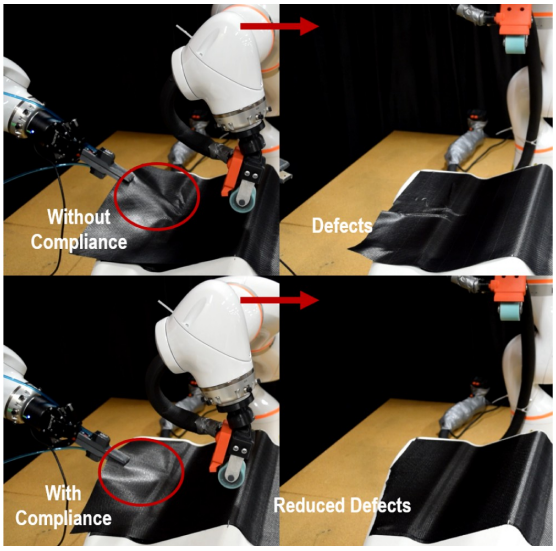

Fig. 36. Physics based simulation has some uncertainty in predicting the real ply state. A high cost ply state is obtained as a result of absence of grasping robot compliance (feedback control) which leads to defects in the layup. The defects can be prevented by using compliance mode for the grasping robots.

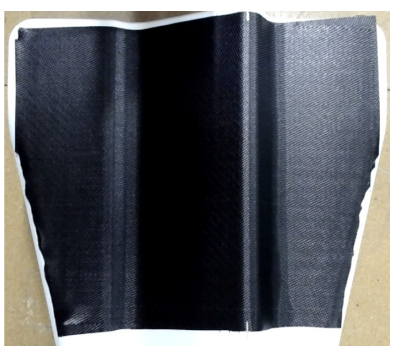

Fig. 37. The completed layup shown on the mold.

Table 9
Time taken by different operations in the entire layup process.

	Time (minutes)	Frequency of operation	Total time (minutes)
Expert Input	14	1	14
Planning time	6.6	1	6.6
Trial run	9.4	2	18.8
Grasp plan modification	3	1	3
Layup	8.5	15	127.5
Human Intervention	1.2	15	18
			187.9

8.5. Discussions

There is uncertainty in the state of the sheet due to environmental factors and material properties. It is challenging to capture these uncertainties in the physics-based simulations. The grasp plans generated needed some modifications as the sheet was coming in contact with the mold. Translating two grasp points corresponding to stage-2 and stage-1 shown in Fig. 35 by 20–30 mm solved the problem. Using a high fidelity simulation and machine learning models to tune the grasp planner parameters can lead to improved solutions. Drape simulation parameters like the bending and shear energy constants also cause inaccuracies in the simulation. A more sophisticated method to tune these parameters by performing experiments and using machine learning can

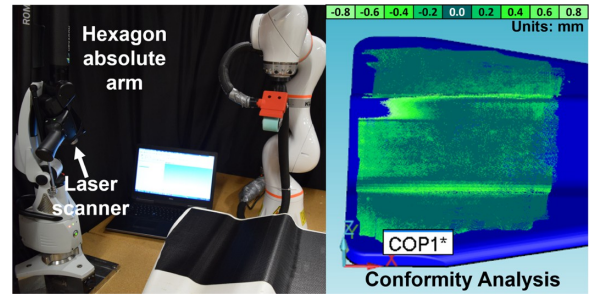

Fig. 38. Conformity analysis using hexagon absolute arm. The color map showing the deviation of the ply with the mold surface is shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

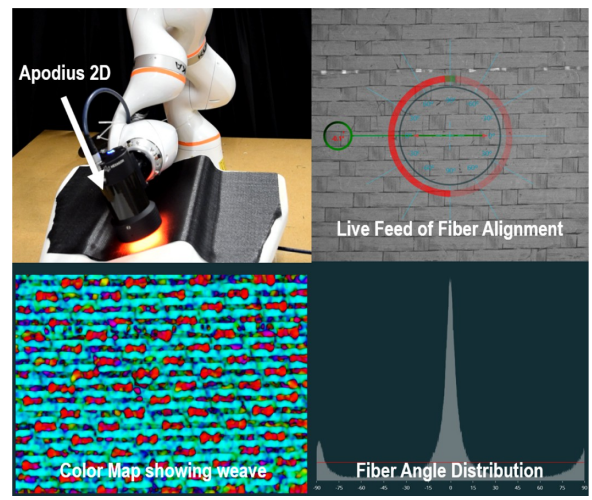

Fig. 39. Apodius 2D sensor attached to the robot flange. Image captured by the sensor at a region of interest. Image is analyzed for determining fiber angle deviation.

produce better grasp plans. End-effector dimensions (e.g. roller width) influence the layup quality and sophisticated research is needed to identify correct dimensions for a given mold geometry. Trials were conducted to find suitable roller dimensions before the layup was perfected. Same grasp plans were used for all the trial runs which shows how using incorrect roller dimensions can introduce defects. Commonly observed defects during the trials were wrinkles, bridging, air pockets, and misalignment. Although, once the grasp plans and parameters are tuned, the process is consistent and several plies can be stacked without observable defects.

9. Conclusions

We addressed the following two major planning challenges towards automation of the composite prepreg layup process: (a) generation of grasp plans and (b) robot placement and path-constrained trajectory generation. We presented a state-space search assisted by physics-based simulations to automate the grasp planning process. This approach successfully generated grasp plans for parts with varying level of complexity. The use of heuristic we developed in this work enabled us to significantly reduce computation time compared to exhaustive search. We developed an approach based on successive application of constraints to solve the robot placement problem. This approach can handle large parts with complex tool paths. The approach also reduces the computation time taken to place the robots for complex part and increases the probability of finding a solution.

In the future, we plan to develop the algorithm which can transfer

the knowledge between the grasp planner and robot placement algorithm. The grasp planner will be improved by taking the draping tool paths into account. The current version of the algorithm decouples the draping and grasping tool paths. The grasp planner generates tool paths that may not be feasible to execute by the robots. Such cases would require an exchange of information between the two searches, and some notion of robot trajectory feasibility will be incorporated in the grasp plans. The robot placement algorithm will be extended to the case where the robots are fixed, and the mold must be placed in the cell. Currently, the placements of the robots are deconflicted by the user, assuming that the robots must be placed around the mold.

CRediT authorship contribution statement

Rishi K. Malhan: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing - original draft, Visualization. **Aniruddha V. Shembekar:** Software, Visualization, Investigation. **Ariyan M. Kabir:** Conceptualization, Investigation, Writing - original draft. **Prahar M. Bhatt:** Software, Investigation, Writing - original draft. **Brual Shah:** Validation, Writing - original draft. **Scott Zanio:** Investigation, Funding acquisition, Visualization. **Steven Nutt:** Conceptualization, Writing - review & editing, Investigation. **Satyandra K. Gupta:** Conceptualization, Methodology, Formal analysis, Funding acquisition, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We dedicate this paper to the memory of the late Dr. Timotei Centea. He was a Research Assistant Professor at the M.C. Gill Composites Center at the University of Southern California. He passed away in March, 2019. He was involved in the development of the robotic cell and this work since the beginning of the project. He has been an influential and inspiring figure for us, and we will continue to celebrate his memory and accomplishments.

This work is supported in part by National Science Foundation Grants #1634431 and #1925084. We acknowledge the software and hardware support from Apodius, a division of Hexagon Manufacturing Intelligence. Alexander Leutner, Abdelrahman Elfar, Patryk Wroclawski, and Branden Smith are the members of the Apodius and Hexagon team who supported the realization of the robotic cell. We acknowledge the software support from the University of Bristol.

References

[1] J. Alonso-Mora, R. Knepper, R. Siegwart, D. Rus, Local motion planning for collaborative multi-robot manipulation of deformable objects, IEEE International Conference on Robotics and Automation (ICRA), (2015), pp. 5495–5502.
[2] APCM, Prepreg Sheet, (2019). (https://www.prepregs.com/da4518/), [Online; accessed August-2019]
[3] N.A. Aspragathos, Optimal location of path following tasks in the workspace of a manipulator using genetic algorithms, Recent Advances in Robot Kinematics, Springer, 1996, pp. 179–188.
[4] N.A. Aspragathos, S. Foussias, Optimal location of a robot path when considering velocity performance, Robotica 20 (2) (2002) 139–147.
[5] Y. Bai, W. Yu, C.K. Liu, Dexterous manipulation of cloth, Comput. Graphics Forum 35 (2) (2016) 523–532.
[6] O.B. Bayazit, J.-M. Lien, N.M. Amato, Probabilistic roadmap motion planning for deformable objects, IEEE International Conference on Robotics and Automation, 2 (2002), pp. 2126–2133 vol.2, https://doi.org/10.1109/ROBOT.2002.1014854.
[7] D. Berenson, Manipulation of deformable objects without modeling and simulating deformation, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2013), pp. 4525–4532, https://doi.org/10.1109/IROS.2013.6697007.
[8] P.M. Bhatt, A.M. Kabir, R.K. Malhan, A.V. Shembekar, B.C. Shah, S.K. Gupta, Concurrent design of tool-paths and impedance controllers for performing area coverage operations in manufacturing applications under uncertainty, IEEE Conference on Automation Science and Engineering, IEEE, 2019.
[9] A. Björnsson, M. Jonsson, K. Johansen, Automated material handling in composite manufacturing using pick-and-place systems–a review, Robot. Comput. Integr. Manuf. 51 (2018) 222–229.
[10] D.M. Bodily, T.F. Allen, M.D. Killpack, Motion planning for mobile robots using inverse kinematics branching, International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 5043–5050.
[11] D.E. Breen, D.H. House, P.H. Getto, A physically-based particle model of woven cloth, Vis. Comput. 8 (5–6) (1992) 264–277.
[12] D.E. Breen, D.H. House, M.J. Wozny, A particle-based model for simulating the draping behavior of woven cloth, Text. Res. J. 64 (11) (1994) 663–685.
[13] R.O. Buckingham, G.C. Newell, Automating the manufacture of composite broadgoods, Compos. Part A 27 (3) (1996) 191–200, https://doi.org/10.1016/1359-835X(96)80001-9.
[14] S. Caro, C. Dumas, S. Garnier, B. Furet, Workpiece placement optimization for machining operations with a kuka kr270-2 robot, IEEE International Conference on Robotics and Automation, (2013), pp. 2921–2926. Karlsruhe, Germany
[15] S. Caro, S. Garnier, B. Furet, A. Klimchik, A. Pashkevich, Workpiece placement optimization for machining operations with industrial robots, IEEE International Conference on Advanced Intelligent Mechatronics, (2014), pp. 1716–1721. Besanon, France
[16] P.Y. Chen, P.M. Popovich, Correlation: Parametric and Nonparametric Measures, Sage, 2002.
[17] J.J. Craig, Introduction to Robotics: Mechanics and Control, 3 Pearson/Prentice Hall Upper Saddle River, NJ, USA, 2005.
[18] J. Das, N. Sarkar, Autonomous shape control of a deformable object by multiple manipulators, J. Intell. Robot. Syst. 62 (1) (2011) 3–27, https://doi.org/10.1007/s10846-010-9436-5.
[19] F. Ding, J. Huang, Y. Wang, T. Matsuno, T. Fukuda, Vibration damping in manipulation of deformable linear objects using sliding mode control, Adv. Rob. 28 (3) (2014) 157–172, https://doi.org/10.1080/01691864.2013.861769.
[20] N.C.N. Doan, W. Lin, Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6r articulated robots, Robot. Comput. Integr. Manuf. 48 (2017) 233–242.
[21] M. Dogar, A. Spielberg, S. Baker, D. Rus, Multi-robot grasp planning for sequential assembly operations, Auton. Robots 43 (3) (2019) 649–664.
[22] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. PetrÃk, A. Kargakos, L. Wagner, V. Hlav, T.K. Kim, S. Malassiotis, Folding clothes autonomously: a complete pipeline, IEEE Trans. Rob. 32 (6) (2016) 1461–1478, https://doi.org/10.1109/TRO.2016.2602376.
[23] C. Dumas, S. Caro, S. Garnier, B. Furet, Workpiece placement optimization of six-revolute industrial serial robots for machining operations, ASME Biennial Conference on Engineering Systems Design and Analysis, (2012), pp. 419–428. Nantes, France
[24] M. Elkington, D. Bloom, C. Ward, A. Chatzimichali, K. Potter, Hand layup: understanding the manual process, Adv. Manuf. 1 (3) (2015) 138–151.
[25] M. Elkington, C. Ward, K.D. Potter, Automated layup of sheet prepregs on complex moulds, SAMPE Long Beach Conference, (2016).
[26] M. Elkington., C. Ward, A. Sarkytbayev, Automated composite draping: a review, SAMPE, SAMPE North America, 2017.
[27] G. Fantoni, M. Santochi, G. Dini, K. Tracht, B. Scholz-Reiter, J. Fleischer, T.K. Lien, G. Seliger, G. Reinhart, J. Franke, et al., Grasping devices and methods in automated production processes, CIRP Ann. 63 (2) (2014) 679–701.
[28] J.T. Feddema, Kinematically optimal robot placement for minimum time coordinated motion, IEEE International Conference on Robotics and Automation, 4 (1996), pp. 3395–3400. Minnesota, USA
[29] S. Flixeder, T. Glck, A. Kugi, Modeling and force control for the collaborative manipulation of deformable strip-like materials, IFAC-PapersOnLine 49 (21) (2016) 95–102, https://doi.org/10.1016/j.ifacol.2016.10.518. 7th IFAC Symposium on Mechatronic Systems MECHATRONICS 2016
[30] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, W. Burgard, Learning the elasticity parameters of deformable objects with a manipulation robot, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2010), pp. 1877–1883, https://doi.org/10.1109/IROS.2010.5653949.
[31] B. Frank, C. Stachniss, N. Abdo, W. Burgard, Efficient motion planning for manipulation robots in environments with deformable objects, 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, (2011), pp. 2180–2185, https://doi.org/10.1109/IROS.2011.6094946.
[32] E. Glorieux, P. Franciosa, D. Ceglarek, Quality and productivity driven trajectory optimisation for robotic handling of compliant sheet metal parts in multi-press stamping lines, Robot. Comput. Integr. Manuf. 56 (2019) 264–275.
[33] Grandviewresearch., Composites market article., 2018, (https://www.grandviewresearch.com/press-release/global-composites-market).
[34] G.G. Gunnarsson, O.W. Nielsen, C. Schlette, H.G. Petersen, Fast and simple interacting models of drape tool and ply material for handling free hanging, pre-impregnated carbon fibre material, International Conference on Informatics in Control, Automation and Robotics, Springer, 2018, pp. 1–25.
[35] S.G. Hancock, K.D. Potter, The use of kinematic drape modelling to inform the hand lay-up of complex composite components using woven reinforcements, Compos Part A 37 (3) (2006) 413–422, https://doi.org/10.1016/j.compositesa.2005.05.044.
[36] J.S. Hemmerle, F.B. Prinz, Optimal path placement for kinematically redundant manipulators, IEEE International Conference on Robotics and Automation, (1991), pp. 1234–1244. Sacramento, CA, USA
[37] D. Henrich, H. Wörn, Robot Manipulation of Deformable Objects, Springer Science

& Business Media, 2012.
[38] Z. Hu, P. Sun, J. Pan, Three-dimensional deformable object manipulation using fast online gaussian process regression, Robot. Autom. Lett. 3 (2) (2018) 979–986.
[39] S.H. Huang, J. Pan, G. Mulcaire, P. Abbeel, Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (2015), pp. 878–885, https://doi.org/10.1109/IROS.2015.7353475.
[40] P. Jiménez, Survey on model-based manipulation planning of deformable objects, Robot. Comput. Integr. Manuf. 28 (2) (2012) 154–163.
[41] S.G. Johnson, The Nlopt Nonlinear-Optimization Package, (2019).
[42] H. Jones, A. Chatzimichali, R. Middleton, K. Potter, C. Ward, Exploring the discrete tools used by laminators in composites manufacturing: application of novel concept, Adv. Manuf. 1 (4) (2015) 185–198.
[43] A.M. Kabir, A. Kanyuck, R.K. Malhan, A.V. Shembekar, S. Thakar, B.C. Shah, S.K. Gupta, Generation of synchronized configuration space trajectories of multi-robot systems, 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 8683–8690.
[44] F.F. Khalil, P. Payeur, Dexterous robotic manipulation of deformable objects with multi-sensory feedback-a review, Robot Manipulators Trends and Development, InTech, 2010.
[45] N. Koganti, T. Tamei, K. Ikeda, T. Shibata, Bayesian nonparametric learning of cloth models for real-time state estimation, IEEE Trans. Rob. 33 (4) (2017) 916–931, https://doi.org/10.1109/TRO.2017.2691721.
[46] P.N. Koustoumpardis, K.I. Chatzilygeroudis, A.I. Synodinos, N.A. Aspragathos, Human robot collaboration for folding fabrics based on force/RGB-D feedback, Advances in Robot Design and Intelligent Control, Springer International Publishing, 2016, pp. 235–243.
[47] C. Krogh, J.A. Glud, J. Jakobsen, Modeling the robotic manipulation of woven carbon fiber prepreg plies onto double curved molds: a path-dependent problem, J. Compos. Mater. 53 (15) (2019) 2149–2164.
[48] D. Kruse, R.J. Radke, J.T. Wen, Collaborative human-robot manipulation of highly deformable materials, IEEE International Conference on Robotics and Automation (ICRA), (2015), pp. 3782–3787, https://doi.org/10.1109/ICRA.2015.7139725.
[49] D. Kruse, R.J. Radke, J.T. Wen, Human-robot collaborative handling of highly deformable materials, American Control Conference (ACC), (2017), pp. 1511–1516, https://doi.org/10.23919/ACC.2017.7963167.
[50] A.M. Ladd, L.E. Kavraki, Using motion planning for knot untangling, Int. J. Rob. Res. 23 (7–8) (2004) 797–808, https://doi.org/10.1177/0278364904045469.
[51] F. Lamiraux, L.E. Kavraki, Planning paths for elastic objects under manipulation constraints, Int. J. Rob. Res. 20 (3) (2001) 188–208, https://doi.org/10.1177/02783640122067354.
[52] A.X. Lee, S.H. Huang, D. Hadfield-Menell, E. Tzeng, P. Abbeel, Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2014), pp. 4402–4407, https://doi.org/10.1109/IROS.2014.6943185.
[53] A.X. Lee, H. Lu, A. Gupta, S. Levine, P. Abbeel, Learning force-based manipulation of deformable objects from multiple demonstrations, 2015 IEEE International Conference on Robotics and Automation (ICRA), (2015), pp. 177–184, https://doi.org/10.1109/ICRA.2015.7138997.
[54] Y. Li, Y. Yue, D. Xu, E. Grinspun, P.K. Allen, Folding deformable objects using predictive simulation and trajectory optimization, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (2015), pp. 6000–6006, https://doi.org/10.1109/IROS.2015.7354231.
[55] Y. Lin, H. Zhao, H. Ding, Posture optimization methodology of 6r industrial robots for machining using performance evaluation indexes, Robot. Comput. Integr. Manuf. 48 (2017) 59–72.
[56] D.C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Math. Program. 45 (1–3) (1989) 503–528.
[57] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, P. Abbeel, Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding, IEEE International Conference on Robotics and Automation (ICRA), (2010), pp. 2308–2315.
[58] R.K. Malhan, R. Jomy Joseph, A.V. Shembekar, A.M. Kabir, P.M. Bhatt, S.K. Gupta, Online grasp plan refinement for reducing defects during robotic layup of composite prepreg sheets, IEEE International Conference on Robotics and Automation (ICRA), (2020). Paris, France
[59] R.K. Malhan, A.M. Kabir, B. Shah, T. Centea, S.K. Gupta, Automated prepreg sheet placement using collaborative robotics, Society for the Advancement of Material and Process Engineering (SAMPE) Technical Conference and Exhibition, (2018).
[60] R.K. Malhan, A.M. Kabir, B. Shah, T. Centea, S.K. Gupta, Determining feasible robot placements in robotic cells for composite prepreg sheet layup, ASMEs 14th Manufacturing Science and Engineering Conference, (2019). Erie, PA, USA
[61] R.K. Malhan, A.M. Kabir, B.C. Shah, S.K. Gupta, Identifying feasible workpiece placement with respect to redundant manipulator for complex manufacturing tasks, IEEE International Conference on Robotics and Automation (ICRA), (2019). Montreal, Canada
[62] R.K. Malhan, A.M. Kabir, A.V. Shembekar, B. Shah, T. Centea, S.K. Gupta, Hybrid cells for multi-layer prepreg composite sheet layup, IEEE 14th International Conference on Automation Science and Engineering (CASE), (2018), pp. 1466–1472.
[63] T. Matsuno, T. Fukuda, F. Arai, Flexible rope manipulation by dual manipulator system using vision sensor, IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No.01TH8556), 2 (2001), pp. 677–682 vol.2, https://doi.org/10.1109/AIM.2001.936748.
[64] D. McConachie, D. Berenson, Bandit-based model selection for deformable object manipulation, arXiv:1703.10254(2017).
[65] D. McConachie, M. Ruan, D. Berenson, Interleaving planning and control for deformable object manipulation, International Symposium on Robotics Research (ISRR), (2017).
[66] S. Miller, J. van den Berg, M. Fritz, T. Darrell, K. Goldberg, P. Abbeel, A geometric approach to robotic laundry folding, Int. J. Rob. Res. 31 (2) (2012) 249–267, https://doi.org/10.1177/0278364911430417.
[67] S. Miller, M. Fritz, T. Darrell, P. Abbeel, Parametrized shape models for clothing, IEEE International Conference on Robotics and Automation (ICRA), (2011), pp. 4861–4868.
[68] S. Mitsi, K.-D. Bouzakis, D. Sagris, G. Mansour, Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm, Robot. Comput. Integr. Manuf. 24 (1) (2008) 50–59.
[69] Modernmachineshop., Managing Risk is this Shop's Bold Strategy. (2018). (https://www.mmsonline.com/articles/managing-risk-is-this-shops-bold-strategy)
[70] R. Molfino, M. Zoppi, F. Cepolina, J. Yousef, E.E. Cepolina, Design of a hyper-flexible cell for handling 3d carbon fiber fabric, Rec. Adv. Mech. Eng.Mech. 165 (2014).
[71] M. Moll, L.E. Kavraki, Path planning for deformable linear objects, IEEE Trans. Rob. 22 (4) (2006) 625–636.
[72] M. Moll, L.E. Kavraki, Path planning for deformable linear objects, IEEE Trans. Rob. 22 (4) (2006) 625–636, https://doi.org/10.1109/TRO.2006.878933.
[73] A. Nektarios, N.A. Aspragathos, Optimal location of a general position and orientation end-effector's path relative to manipulator's base, considering velocity performance, Robot. Comput. Integr. Manuf. 26 (2) (2010) 162–173.
[74] B. Nelson, M. Donath, Optimizing the location of assembly tasks in a manipulator's workspace, J. Robot. Syst. 7 (6) (1990) 791–811.
[75] G. Newell, K. Khodabandehloo, Modelling flexible sheets for automatic handling and lay-up of composite components, Proc. Inst. Mech. Eng. Part B 209 (6) (1995) 423–432.
[76] J. Nocedal, Updating quasi-newton matrices with limited storage, Math. Comput. 35 (151) (1980) 773–782.
[77] A. Papacharalampopoulos, S. Makris, A. Bitzios, G. Chryssolouris, Prediction of cabling shape during robotic manipulation, Int. J. Adv. Manuf. Technol. 82 (1–4) (2016) 123–132.
[78] S. Patil, et al., Motion planning under uncertainty in highly deformable environments, Robot. Sci. Syst. (2011).
[79] Z. Peng, L. Yuanchun, Position/force control of two manipulators handling a flexible payload based on finite-element model, IEEE International Conference on Robotics and Biomimetics (ROBIO), (2007), pp. 2178–2182, https://doi.org/10.1109/ROBIO.2007.4522507.
[80] S. Rodriguez, J.-M. Lien, N.M. Amato, Planning motion in completely deformable environments, IEEE International Conference on Robotics and Automation, (2006), pp. 2466–2471, https://doi.org/10.1109/ROBOT.2006.1642072.
[81] O. Roussel, A. Borum, M. Taix, T. Bretl, Manipulation planning with contacts for an extensible elastic rod by sampling on the submanifold of static equilibrium configurations, IEEE International Conference on Robotics and Automation (ICRA), (2015), pp. 3116–3121.
[82] M. Saha, P. Isto, Motion planning for robotic manipulation of deformable linear objects, IEEE International Conference on Robotics and Automation, (2006), pp. 2478–2484, https://doi.org/10.1109/ROBOT.2006.1642074.
[83] M. Saha, P. Isto, Manipulation planning for deformable linear objects, IEEE Trans. Rob. 23 (6) (2007) 1141–1150, https://doi.org/10.1109/TRO.2007.907486.
[84] D.W. Sandberg, R.B. Wodtli, Collision detection using sphere approximations, Robotics and Factories of the Future, Springer, 1988, pp. 456–460.
[85] J. Schulman, J. Ho, C. Lee, P. Abbeel, Learning from Demonstrations Through the Use of Non-rigid Registration, Springer International Publishing, Cham, pp. 339–354. 10.1007/978-3-319-28872-7_20.
[86] G. Seliger, F. Szimmat, J. Niemeier, J. Stephan, Automated handling of non-rigid parts, CIRP Ann. 52 (1) (2003) 21–24.
[87] P. Tsarouchi, J. Spiliotopoulos, G. Michalos, S. Koukas, A. Athanasatos, S. Makris, G. Chryssolouris, A decision making framework for human robot collaborative workplace generation, Procedia CIRP 44 (2016) 228–232.
[88] D. Tzeranis, Y. Ishijima, S. Dubowsky, Manipulation of large flexible structural modules by space robots mounted on flexible structures, Proc. Int. Sym. on Artificial Intelligence, Robotics and Automation in Space, (2005).
[89] R. Ur-Rehman, S. Caro, D. Chablat, P. Wenger, Multi-objective path placement optimization of parallel kinematics machines based on energy consumption, shaking forces and maximum actuator torques: application to the orthoglide, Mech. Mach. Theory 45 (8) (2010) 1125–1141.
[90] N. Vahrenkamp, T. Asfour, R. Dillmann, Robot placement based on reachability inversion, IEEE International Conference on Robotics and Automation, (2013), pp. 1970–1975. Karlsruhe, Germany
[91] J. Van Den Berg, S. Miller, K. Goldberg, P. Abbeel, Gravity-based robotic cloth folding, Algorithmic Foundations of Robotics IX, Springer, 2010, pp. 409–424.
[92] H. Wakamatsu, E. Arai, S. Hirai, Knotting/unknotting manipulation of deformable linear objects, Int. J. Rob. Res. 25 (4) (2006) 371–395, https://doi.org/10.1177/0278364906064819.
[93] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, T. Ogata, Repeatable folding task by humanoid robot worker using deep learning, Robot. Autom. Lett. 2 (2) (2017) 397–403.
[94] T. Yoshikawa, Analysis and control of robot manipulators with redundancy, Robotics Research: The First International Symposium, MIT press Cambridge, MA, 1984, pp. 735–747.
[95] T. Yoshikawa, Manipulability of robotic mechanisms, Int. J. Rob. Res. 4 (2) (1985) 3–9.

[96] T. Yoshikawa, Foundations of Robotics: Analysis and Control, Mit Press, 1990.
[97] F. Zacharias, Knowledge Representations for Planning Manipulation Tasks, 16 Springer Science & Business Media, 2012.
[98] F. Zacharias, C. Borst, G. Hirzinger, Capturing robot workspace structure: representing robot capabilities, IEEE International Conference on Intelligent Robots and Systems, (2007), pp. 3229–3236. San Diego, CA, USA
[99] F. Zacharias, W. Sepp, C. Borst, G. Hirzinger, Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories, IEEE International Conference on Humanoid Robots, (2009), pp. 55–61. Paris, France
[100] P. Zhang, Y. c. Li, Simulations and trajectory tracking of two manipulators manipulating a flexible payload, IEEE Conference on Robotics, Automation and Mechatronics, (2008), pp. 72–77, https://doi.org/10.1109/RAMECH.2008.4681323.