Towards the automation of woven fabric draping via reinforcement learning and Extended Position Based Dynamics

Patrick M. Blies, Sophia Keller, Ulrich Kuenzer, Yassine El Manyari, Franz Maier, Markus G. R. Sause, Marcel Wasserer, Roland M. Hinterhölzl

Angaben zur Veröffentlichung / Publication details:

Blies, Patrick M., Sophia Keller, Ulrich Kuenzer, Yassine El Manyari, Franz Maier, Markus G. R. Sause, Marcel Wasserer, and Roland M. Hinterhölzl. 2025. "Towards the automation of woven fabric draping via reinforcement learning and Extended Position Based Dynamics." Journal of Manufacturing Processes 141: 336–50. https://doi.org/10.1016/j.jmapro.2025.02.063.

Journal of Manufacturing Processes 141 (2025) 336–350

Contents lists available at ScienceDirect

Journal of Manufacturing Processes

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

Research article

Towards the automation of woven fabric draping via reinforcement learning and Extended Position Based Dynamics

Patrick M. Blies [a], Sophia Keller [b,*], Ulrich Kuenzer [c], Yassine El Manyari [a], Franz Maier [b], Markus G.R. Sause [d], Marcel Wasserer [a], Roland M. Hinterhölzl [b]

[a] enliteAI GmbH, Wollzeile 24/12, Vienna, 1010, Vienna, Austria
[b] University of Applied Sciences Upper Austria, Stelzhamerstraße 23, Wels, 4600, Upper Austria, Austria
[c] Alpex Technologies GmbH, Gewerbepark 38, Mils, 6068, Tyrol, Austria
[d] University of Augsburg, Universitätsstrasse 2, Augsburg, 86159, Bavaria, Germany

ARTICLE INFO

Keywords:
Surrogate model
Reinforcement learning
Draping simulation
Composite preforming
Extended Position Based Dynamics
Artificial Intelligence
Digital twin

ABSTRACT

The draping process in the preforming stage of composite manufacturing is very cost- and time-expensive and requires substantial manual labor. One strategy towards automation is the use of collaborative robots. Recent advances in AI have made it possible to train robots on difficult real-world tasks with reinforcement learning. However, training a robot using reinforcement learning is practically challenging and leveraging simulation is often the only option to use reinforcement learning in real-world settings at all. Existing FE models, which are commonly used for optimization of preforming processes, are too slow for reinforcement learning training. We addressed this issue by developing an XPBD-based surrogate model, drastically reducing simulation times compared to a classic FE model. With the achieved speedup, the training of a reinforcement learning agent became feasible and a draping trajectory could successfully be transferred to a real-world cobot, demonstrating the potential and capabilities of this innovative approach.

1. Introduction

In the composite industry, about one third of carbon fiber reinforced components are manufactured using a draping procedure to obtain the component geometry [1]. Due to their extraordinary properties, the amount of composite components used in the aircraft industry, and thereby the amount of draped components, is increasing steadily [2]. The draping step is typically part of the preforming stage, where a semifinished product (e.g., dry woven fabrics, prepregs or non-crimp fabrics (NCFs)) is placed onto a tool and prefixed using some adhesive agent or binder. This layup process often requires many manual steps, making the preforming phase very cost- and time-expensive [3]. To reduce the expenses in the manufacturing process, it is therefore essential to address the draping step in the process chain. One strategy close at hand to optimize the draping step is to reduce manual work by automating this process step.

In industry, (semi-)automated draping methods such as automated fiber placement (AFP), automated tape laying [4] and diaphragm-forming [5] are already well-established for manufacturing preforms for aircraft components. They are commonly used for simple geometries such as single-curved panels or C- and L-profiles for stiffening ribs. However, at present, the amount of manual labor scales with geometry complexity. This is mainly because complex components require higher flexibility and sensitivity in the layup process, which cannot be fully achieved by conventional automated processes. Therefore, latest research focuses on the development of more flexible solutions for automated composite manufacturing [6–8]. One way of increasing flexibility in the automated manufacturing is to develop new manufacturing processes using collaborative robots (cobots), which are designed to perform precise tasks and are able to mimic manual processing steps [9,10].

New composite manufacturing processes are typically developed by experts in an iterative process where an initial execution environment is set up in a CAD software, implemented in real-world and optimized using trial-and-error experiments. This process is accompanied by numerical process simulations using a digital twin in a finite element (FE) software. For semifinished products such as dry fabrics or prepregs, however, the simulations can have substantial calculation times [11]. This inhibits an efficient process optimization and determination of optimum parameters, requiring a lot of expertise by qualified workers to keep the development effort to a minimum.

* Corresponding author.
E-mail address: sophia.keller@fh-wels.at (S. Keller).

https://doi.org/10.1016/j.jmapro.2025.02.063
Received 10 October 2024; Received in revised form 14 January 2025; Accepted 24 February 2025
Available online 6 March 2025

A commonly used strategy to tackle that issue is surrogate-based optimization (SBO), where simulation processes are approximated using computationally inexpensive operations. Especially in sheet metal forming, SBO is already an established method [12,13]. Compared to sheet metal, the reduced stiffness and slack forming behavior of composite preforms poses an extra challenge to solving the problem. Therefore, only in the last few years, SBO has been established within that field. Recent studies focus on using artificial intelligence (AI) to learn surrogate models in order to facilitate and speed up the optimization of manufacturing processes [14]. These AI-based surrogate models have also been applied to approximate the behavior of FE simulation models and enable faster calculation times [15–18]. However, the technology readiness level (TRL) of surrogate models for composite manufacturing – whether AI-based or not – is still quite low, which is why the developed surrogates are used on a laboratory scale rather than in industry.

Another field where fast simulations of physical processes are required is physics-based animation. The simulation field was introduced to graphics by Terzopoulos et al. [19] in 1987. It ranges from the simulation of solid objects, soft bodies and cloths up to fluids, fire and smoke simulations. The difference in simulations between the computational sciences and computer graphics lies in the importance of accuracy: computational sciences seek to replace real-world experiments and should be as accurate as possible. The main application of simulations in computer graphics are visual effects for movies and games that should look realistic. The focus is much more on speed and controllability, and much less on accuracy. Hence, the computer graphics community has produced methods and approaches that are much more efficient than traditional methods, while not really focusing on accuracy. One such method is Position Based Dynamics (PBD) [20], which — besides applications in computer graphics, such as [21,22] — has also been applied in other fields, for instance in medicine [23–25] and robotics [26,27].

Reinforcement learning (RL) is a method that has recently seen a lot of success in solving sequential decision making problems across a lot of different subfields, such as manufacturing [28,29], medicine [30,31], robotics [32], communications [33] and energy [34]. Since draping path optimization is also a sequential decision making problem, RL can be applied here. One downside when applying RL to a real-world problem is that you usually require an efficient simulation of the problem you want to solve. An FE model is not suitable for this since its runtime for a single simulation is simply too long. This is why we developed a PBD-based surrogate model to serve as the simulation backbone in the RL optimization pipeline in this work. As it turned out, this surrogate is fast enough to enable the use of RL for the draping optimization problem.

The aim of this study is to investigate the basic feasibility of using reinforcement learning (RL) in combination with PBD to create optimized work instructions for an automated draping process with a robot. For this purpose, we created an FE draping simulation model, which serves as our baseline. The model includes a draping tool geometry and the mechanical data of a glass-fiber woven fabric obtained by standard material characterization methods. We validated the FE baseline by comparing the results of experimental and numerical diaphragm-draping using the characterized material. Subsequently, we developed a PBD-based surrogate of the FE model. In contrast to AI-based surrogates, this is generalizable to any geometry. By comparing the surrogate to the FE model, we ensure that the surrogate yields a sufficient approximation of the baseline results. Optimized work instructions were determined by using RL in combination with the PBD-based surrogate simulation. We defined a test case for which the developed method determines an optimal draping trajectory. The trajectory was transferred to a custom real-world execution environment to validate the AI-proposed draping strategy for our test case.

Our work is structured in the following way: we describe the characterization of the material we used in Section 2.1 and introduce the FE model using the characterized material in Section 2.2. The PBD-based surrogate model is described in Section 3. The RL approach to determine an optimal draping path is introduced in Section 4 and the method for comparing FE and PDB-based model is described in Section 5. The real-world execution environment is described in Section 6. The results are presented in Section 7 and we draw our conclusions in Section 8. Potential future work is discussed in Section 9.

2. FE digital twin

To create a simulation model that provides the baseline for draping results, we created a digital twin in the commercial finite element software Abaqus/CAE [35].

2.1. Material

For complex and highly nonlinear problems involving large deformations and velocities (such as in a draping process), a dynamic, explicit FE solver is required. Especially for slack fabrics, this causes increased calculation times because the time-determining factor for such simulations is the stable time increment Δt_{stable}. Per definition, the stable time increment ($\Delta t_{stable} = L^e/c_d$) depends on the smallest element length L^e of the FE mesh and the wave speed c_d of the material. The wave speed ($c_d = \sqrt{E/\rho}$), in turn, is defined by the ratio of the Young's modulus E to the mass density ρ of the material [35].

Therefore, we selected the glass fiber woven fabric *Interglas 92125* with a 2 × 2 twill weave pattern and a ply thickness of about 0.35 mm for the lamina. Compared to carbon fiber materials (which are commonly used for structural aircraft components), the E to ρ ratio of glass fiber materials is smaller, leading to an increased stable time increment.

2.1.1. Material characterization

To obtain mechanical data for the numerical simulation, we characterized the material using the following testing methods:

Uniaxial Extension. We used the uniaxial test setup as described in ISO 13934-1 to test the mechanical response in warp and weft direction of the fabric. In each direction, 6 specimens of 200×60 mm were tested using the Zwick/Roell Z020 (20 kN load cell) with a testing velocity of 0.33 mm/s.

In-Plane Shear. We tested the in-plane shear response using the bias-extension test as described in [36]. Samples (170 × 50 mm) with a +45° (n=10) and −45° (n=11) angle shift to the warp direction were prepared for testing. We used the Zwick/Roell Z0.5 with a testing velocity of 1.6 mm/s.

Bending. To determine the bending behavior of the fabric, we used a manual cantilever bending test apparatus as shown in [37]. As for the uniaxial extension test, we tested 6 specimens (150 × 20 mm) in each direction. The bending length was measured using a ruler.

Friction. The friction between tool and fabric was tested using a testing setup inspired by ISO 8295 with a carrier system (200 g sled) [37]. We tested the friction between the specimens (6 samples per direction) and a steel sheet with the same surface property as the draping tool.

2.1.2. Validation of characterized material

To validate the characterized material data, we conducted experimental draping tests using a diaphragm forming process and compared them to the simulation result. Therefore, we modified the setup described in [38] by replacing the tool with a hemisphere tool and using a single ply of the characterized glass fiber material. The dry fabric was fixed on the tool using a spray adhesive.

We conducted 11 draping experiments and compared the results with the simulation result for our material. For the comparison, we analyzed the topology, the boundary contour and the fiber orientations in warp and weft directions. The topology and boundary contour of the experiments were captured using a 3D laser scanner and Geomagic Control X and the fiber orientations were captured using a fiber angle sensor (FScan, Profactor).

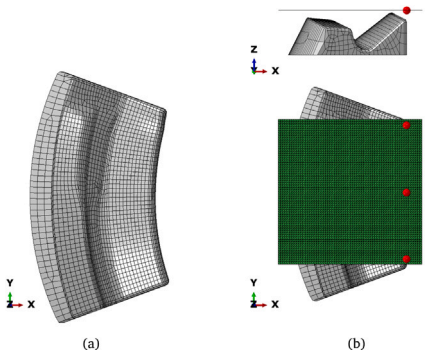


Fig. 1. (a) Meshed rib tool and (b) setup of the FE model before draping, with quadratic blank cut placed above the tool, fixated at the red spherical areas.

2.2. FE model setup

Fig. 1 shows the FE model setup before draping. In the model, a quadratic lamina blank cut (370×370 mm) was placed above a double-curved rib tool (Fig. 1a) and fixed at three clamping points (red spheres in Fig. 1b). The lamina blank cut was meshed using 8192 S3 elements. Since our surrogate model has its root in computer graphics, where it is common to work with triangular meshes, triangular elements were preferred over quad elements. We implemented the characterized fabric material using the hyperelastic, test-data based *FABRIC material model in Abaqus.

Using this FE model, different draping scenarios can be realized. Starting from an initial draped geometry (i.e., a predefined draping use case containing topological draping defects such as wrinkles and bridging), local pressure loads can be applied on the top surface of the lamina to smooth out draping defects.

To simulate continuous load application and enable the realization of different draping paths, we created a VDLOAD subroutine. Within the subroutine, loads are defined within a rectangular bounding box, mimicking a load application using a roller. The bounding box is defined in 2D and projected onto the lamina geometry. In each time step, the bounding box can be shifted in space to imitate continuous loading. The pressure load is set to 1 $\mathrm{N/mm^2}$. Areas which are already draped remain at 1 $\mathrm{N/mm^2}$, simulating the fixation of the lamina onto the tool using a spray-adhesive.

3. XPBD surrogate model

Having the FE model in place, the next step was to increase its efficiency up to the point where an interactive draping approach with reinforcement learning is possible. Classical methods to increase the efficiency of an FE model are to reduce the mesh resolution (i.e., increasing the element size) and (in case of shell elements) to reduce the number of integration points through thickness. Another effective measure is to use mass scaling, where the mass of some elements is increased to augment the stable time increment. Similarly, it is also possible to reduce the time of dynamic events (time scaling). However, both the mass and time scaling can lead to unrealistic inertial or dynamic effects. Overall, the above-mentioned measures are always accompanied by a loss in output precision. More general approaches, that do not influence output precision, are to reduce the output frequency to a minimum and to use symmetry (as far as possible) [39]. Likewise, a quite common approach that does not influence the simulation result is to use CPU parallelization in order to distribute the computational workload across multiple CPU cores, speeding up the calculation process. Although parallelization enables a considerable speedup, it does have limitations and its effect depends on the type of problem.

For the application of RL, calculation times must be within an interactive range. Considering that calculation times for FE draping simulations are typically ranging from several hours to days [11], a speedup of that magnitude is unrealistic for conventional acceleration methods. Therefore, to achieve the required speedups for RL, other approaches must be investigated. One option is to go with a learning-based approach. This has the drawback of a limited generalization capability and usually requires a large amount of training data. In addition, for a draping problem, the correct representation of contact forces is essential, which intensifies the need for sufficient training data for any geometry.

Instead, the approach we investigated for this work was to develop a manual surrogate model for the lamina based on a method that has been used in computer graphics community for a long time and is known for its efficiency: Position Based Dynamics (PBD) [20] and its enhancement Extended Position Based Dynamics (XPBD) [40]. Compared to a learned model, a simulation approach has the advantage of easier generalization to unseen geometries because once a proper surrogate model for the lamina has been developed, this lamina surrogate can be used in an interactive way with any tool geometry. In addition, it does not require any data to train on and handles contact forces naturally.

3.1. Extended position based dynamics

XPBD is a simulation technique that originates in computer graphics and physics-based animation. It belongs to the class of position based simulation methods. To give an intuition as to why XPBD is more efficient than a traditional simulation method, we give a very short comparison of the two methods:

Classical simulation. Classical dynamics simulation methods typically formulate the change of momentum of a system as a function of the applied forces: internal and external forces are accumulated, from which accelerations are computed based on Newton's second law. For a system of n particles with positions $\boldsymbol{x}_i$, this amounts to solving the n equations

$$\begin{aligned} \dot{\boldsymbol{v}}_i &= \frac{1}{m_i}\boldsymbol{f}_i, \quad i \in [1..n] \\ \dot{\boldsymbol{x}}_i &= \boldsymbol{v}_i \end{aligned} \tag{1}$$

where $\boldsymbol{f}_i$ is the sum of all forces acting on particle i. Now, for realistic physical systems, the calculation of internal forces can be very complex, often relying on constitutive equations and the resulting differential equations can be hard to solve numerically. Furthermore, if the differential equations are stiff, taking too large timesteps during time integration can lead to instabilities, which means that the achievable timestep can be severely limited by the stiffness properties of the equation.

Discretization of the differential equations with FE method leads to a large and often sparse system of equations, which needs to be solved at each time step. This is computationally expensive and direct solvers typically have a time complexity of $O(n^3)$, while iterative solvers still have at least $O(n \log n)$, where n is the number of particles used to model the system.

XPBD. Position based approaches, on the other hand, omit the velocity and acceleration layers for the internal forces $\boldsymbol{f}_i$ and compute positions directly by iteratively solving a set of geometric constraints, which describe the desired properties of the simulated system, such as the residual distance between two particles (see Fig. 2 for an example of constraints). This has the advantage that a computation of the internal forces is not necessary and we get no large linear system of equations that needs to be solved. Instead, each constraint can be processed independently, which makes the process highly parallelizable on a GPU, making it suitable for real-time applications such as games or interactive simulations.

Furthermore, XPBD is generally more stable and less prone to numerical instabilities, in particular for stiff systems or large time steps.

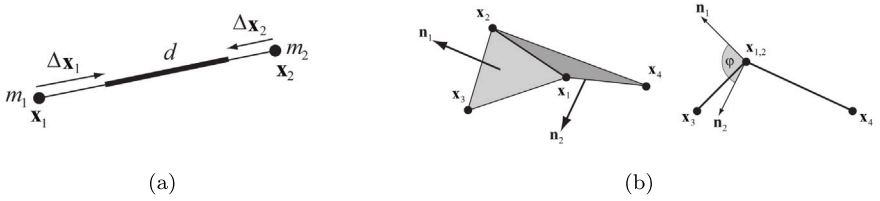


Fig. 2. Depiction of the stretching and bending constraints: (a) Projection of the distance constraint $C(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\| - d$. $\mathbf{x}_i$ is the position of vertex i, m_i is its mass and d is the initial distance of the two vertices and (b) The bending resistance is modeled with the constraint function $C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \phi_0$. $\mathbf{x}_i$ is the position of vertex i, $\mathbf{n}_j$ is the normal of face j. φ_0 is the initial, φ is the actual dihedral angle between the normals of two faces. $\mathbf{x}_{1,2}$ is shorthand for $\mathbf{x}_1 - \mathbf{x}_2$ From [41].

This means that the time integration step for stiff systems can be larger than with traditional methods.

The combination of not having to compute internal forces but instead being able to use a parallelizable constraint projection step and using larger time steps makes XPBD able to achieve a speedup in simulation performance that we need for RL.

Of course, the tradeoff for the speedup that XPBD provides is a limited accuracy. In graphics and animation, the accuracy is not of primary importance since the only goal is visual plausibility. In this work, we investigated whether this reduced accuracy is still sufficient when using XPBD as surrogate when draping composites in an RL pipeline.

For a more thorough introduction into PBD and XPBD, the reader is referred to the survey by Bender et al. [41].

3.2. Our implementation

To develop the XPBD surrogate, we built an interactive simulation environment from scratch with the Taichi programming language [42]. The advantage of the Taichi language is its combination of simple Python syntax, efficient use of GPUs and inbuilt rendering support, which makes it ideally suited for prototyping and studying novel approaches.

3.2.1. Representation of the lamina

We used the same mesh for the surrogate model as for the FE model, i.e., a 370 × 370 mm triangle mesh with 8192 faces and 4225 vertices. We represented the mesh as a particle system, where each mesh vertex corresponds to a particle with position $\mathbf{x}_i$ and mass m_i. Depending on the degree of each vertex, the weight is distributed to attribute to the fact that edge and boundary vertices have different structural contributions than inside vertices. For our mesh, this gives particle masses in the range of 0.0049 g to 0.029 g.

As PBD constraints for the lamina, we used distance constraints (see Fig. 2(a)) and bending constraints (see Fig. 2(b)). The bending compliance was manually adjusted to achieve a high similarity with the FE model.

3.2.2. Representation of the tool

For the tool, we used the same mesh as the FE model. It had 4575 vertices and 9146 triangular faces and was treated as a static collision object.

3.2.3. The XPBD solver

We used the substep XPBD algorithm that was presented in [43]. The algorithm is reprinted in Algorithm 1 for the reader's convenience. For the solution of the constraint equations, we used a parallelized, colored Gauss–Seidel method.

Algorithm 1 Substep XPBD simulation loop from [43]

perform collision detection using $\mathbf{x}^n$, $\mathbf{v}^n$
$\Delta t_s \leftarrow \frac{\Delta t_f}{n_{\text{steps}}}$
while $n < n_{\text{steps}}$ **do**
  predict position $\tilde{\mathbf{x}} \leftarrow \mathbf{x}^n + \Delta t_s \mathbf{v}^n + \Delta t_s^2 \mathbf{M}^{-1} \mathbf{f}_{\text{ext}}(\mathbf{x}^n)$
  for all constraints **do**
    compute $\Delta\lambda$
    compute $\Delta\mathbf{x}$
    update $\lambda^{n+1} \leftarrow \Delta\lambda$ (optional)
    update $\mathbf{x}^{n+1} \leftarrow \Delta\mathbf{x} + \tilde{\mathbf{x}}$
  end for
  update velocities $\mathbf{v}^{n+1} \leftarrow \frac{1}{\Delta t_s}(\mathbf{x}^{n+1} - \mathbf{x}^n)$
  $n \leftarrow n + 1$
end while

3.2.4. Interacting with the simulation

We added an interactive component to the simulation, such that is possible to apply a force to any position of the lamina. This can be done either as a human interactively with the mouse or as an AI agent programmatically. When the simulation is controlled by humans, they can choose the position where the lamina should be draped. The tip of the mouse pointer is taken as the middle position for the interaction. Around this middle position, a stencil is calculated that approximates the size and behavior of the actuator of the real robot (see Section 6).

For this initial proof of concept, we have chosen not to simulate the robot arm itself. While this has certain drawbacks when transferring the determined paths from the simulation to the real robot later in the pipeline (see Section 7.4), it significantly simplifies the simulation of modeling the whole robot just by its actuator. We opted for discrete loading in the simulation, meaning that the stencil is pressed down and lifted again before it is pressed down at the next position. We approximated the contact surface between the end-effector and the tool as a square of size 18×18 mm^2. With our mesh resolution, the stencil typically covers 9 vertices. Fig. 8 shows a visualization of the stencil.

The stencil is always rotated such that its normal is parallel to the tool face normal that is nearest to the mouse ray, to ensure that the draping force is applied along the normal direction of the tool surface. Fig. 3 shows how the stencil changes orientation depending on the normal surface.

Once a vertex has been acted upon by a stencil, it counts as draped when the distance to the tool surface lies within a given threshold. In cannot be moved any more once it has been draped. This represents the behavior of applying a spray adhesive to the lamina, which was done for the real-world setup (see Section 6.1).

Fig. 4 shows a visualization of a manually draped trajectory and the surrogate behavior for this path.

3.2.5. Simulation efficiency

The surrogate needs to be reasonably fast when using it as part of an RL pipeline. On the development laptop, which runs on an Intel Core i7-10870H CPU @ 2.20 GHz and an NVIDIA GeForce RTX 3070, the surrogate simulation runs with about 20–30 FPS. This means

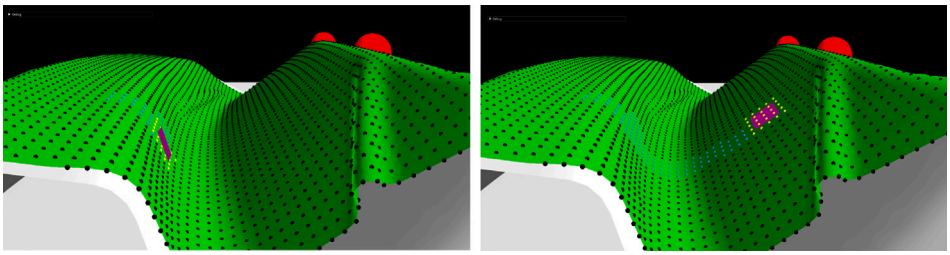


Fig. 3. Visualization of applying the draping force in the surrogate simulation. The purple rectangle shows the area and direction how the force is applied to the surface. The area of the square is 18×18 mm^2. The yellow vertices show the action mask (see Section 4.2.1) for the RL agent. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

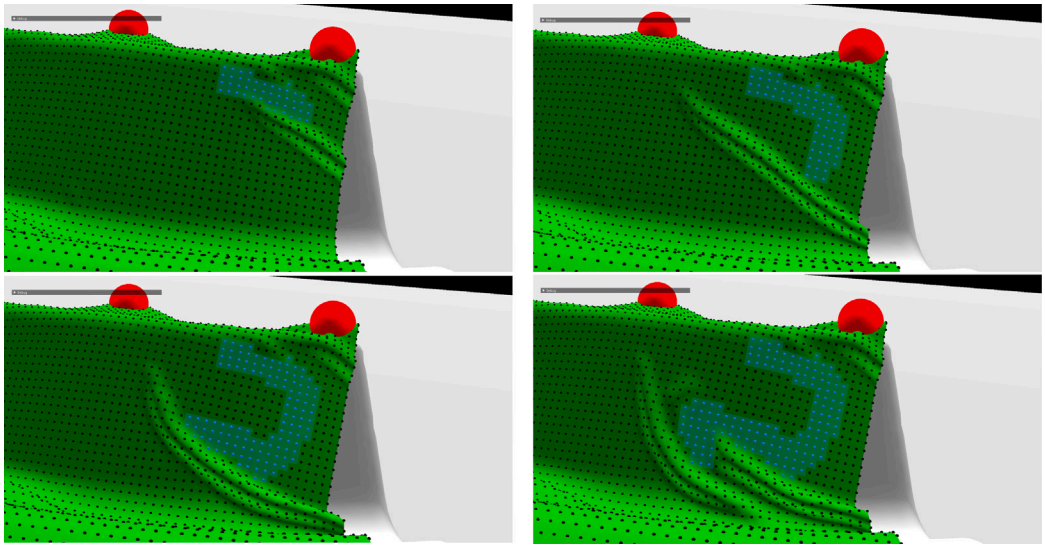


Fig. 4. Visualization of a manually draped trajectory in our simulation environment.

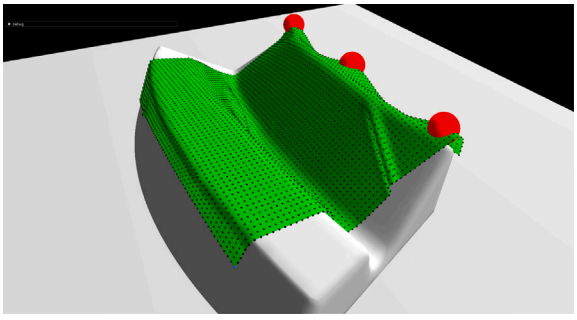
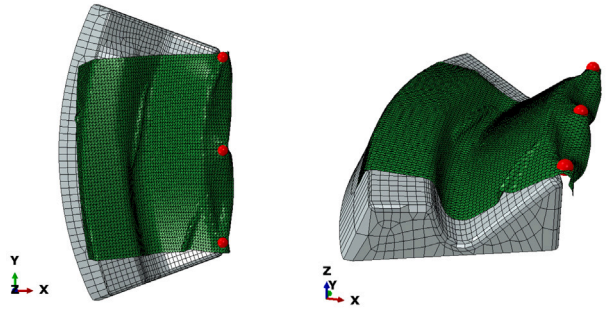


Fig. 5. Isometric view of the initial state of the surrogate simulation model.



(a) (b)

Fig. 6. (a) Top view and (b) isometric view of the initial condition in the FE environment for the draping path determination.

that one simulation step takes at most 0.05 s. In this regime, optimization with RL is possible. A more detailed comparison of the surrogate performance and the FE model performance can be found in Section 7.2.2.

3.3. Testcase setup

To obtain an initial draping state to serve as our test case, we have attached the lamina with three clamps on one side and then simulated a free fall onto the tool. The resulting lamina geometry contains two major wrinkles and serves as the test case for this work. Fig. 5 shows the starting configuration in our simulation environment.

In order to create identical initial conditions for both the XPBD and the FE simulation model, we transferred the deformed lamina mesh back to the FE simulation model (see Fig. 6).

4. RL-based draping path optimization

Having the surrogate simulation in place, we could turn towards solving the draping problem that we started with. We initialized our surrogate simulation with the initial condition visible in Fig. 6 and started the training of the RL agent. The objective of the agent was to find a path of stencil application points such that the lamina is fixed to the surface of the tool and no wrinkles are present (see Fig. 13 for depiction of start and desired goal state).

Initially, our goal was to have the agent drape the whole laminate. This turned out to be infeasible for our hardware and time constraints. That is why we introduced a focus region: we basically simplified the

experiment by limiting the agent's action space to the area around one of the two wrinkles. The objective with the active focus region was then to find a trajectory of stencil application points, such that all vertices in the focus region are fixed to the surface and the wrinkles are gone.

The RL agent had the same boundary conditions that were described in Section 3.2.4, i.e., once a vertex was touched by a stencil, it cannot be moved any more and will stay fixed to the surface until reset of the trial. Initially, when the agent was still randomly exploring, this led to many unsolvable states of the environment because the agent pressed down and thereby fixed the laminate at the wrong locations, making it impossible to drape out the wrinkle.

We will now give a more formal description of the process.

4.1. MDP formulation

In essence, the problem is a sequential decision making problem because with each step in the draping process, the lamina geometry changes (by defect formation, e.g.) such that for the next step, the changed geometry needs to be taken into account. One way to formalize sequential decision making problems is the Markov Decision Process (MDP) [44] and one method to solve MDPs is reinforcement learning.

In general, the components of an MDP are the following:

- a set of states (the state space)
- a set of actions (the action space)
- formulation for the reward obtained after transitioning from one state to another state due to an action
- transition probabilities that determine how one state leads to another state or a simulation of the process

In this section, we describe how we approached each of these for the draping problem.

4.1.1. State space

The state of the system was encoded as an image enabling the use of an efficient, real-time capable class of models: Convolutional Neural Networks (CNNs) [45].

We encoded the state in the following way: the mesh of the lamina is square and has 65×65 vertices. For the state encoding, we represented each vertex as a single pixel in an image of size $65 \times 65 \times 3$. The channels contain the following information:

- The first channel contains binary information about whether the vertex has already been draped to the surface. A vertex is defined as fixed to the surface when it has been pressed down by the stencil and came into contact with the collision object.
- The second channel encodes the distance of the respective vertex to the collision object as real value.
- The third channel contains binary information about whether the stencil has already attempted to push down the vertex.

Mathematically, the state space can be expressed as

$$S = \{\mathbf{s} \in \mathbb{I}^{65\times65} \times \mathbb{R}^{65\times65} \times \mathbb{I}^{65\times65}\},$$

where $\mathbb{I} = \{0, 1\}$ represents the binary values.

Two example states are shown in Fig. 7. Fig. 7(a) shows the encoded initial state, visible in Fig. 5. Only one wrinkle is visible because we introduced focus regions (see Section 4.2.2).

4.1.2. Action space

We defined the action space as the set of integers between zero and the number of vertices in the cloth mesh (4225 in our case) minus 1:

$$A = \{0, 1, 2, \dots, 4224\}$$

After choosing an action, the same stencil that is used for the manual interaction (see Section 3.2.4) is calculated around the chosen vertex that is used for the interaction. This action space will be significantly reduced with the design of an action masking approach (Section 4.2.1).

4.1.3. Reward formulation

We chose a sparse reward formulation that only provides reward to the agent at the end of an episode. One episode is defined as a single trial of the agent, i.e., it starts with 0 vertices touched and ends when every vertex has been touched at least once. After one episode, the environment is reset to the initial position.

The exact reward depends on the focus region (see Section 4.2.2) and is defined the following way: at the end of an episode, the distance of each vertex of the lamina to the tool is computed. Then, all the distances are summed and the reward is calculated by

$$r = D_{\text{init}} - \sum_{j=0}^{4224} d_j.$$

D_{init} is the initial sum of all vertex distances for the currently used focus region at the start of the episode (which is constant across all episodes for a given initial condition and focus region) and d_j is the distance of lamina vertex j to the tool. The reward is only calculated at the end of each episode. For each intermediate step, the provided reward is 0.

4.1.4. Transition probabilities

This is where our simulation surrogate comes into play. Usually, an MDP requires the specification of the probabilities defining how new states develop from previous states and actions,

$$p(s_{t+1}|s_t, a_t).$$

When a simulator is available, this can be left unspecified because the probability is then approximated by sampling experience tuples from the simulator. For us, the surrogate simulation serves as the simulator and provides us with a new state when we supply it with an action.

4.2. Reducing the exploration space

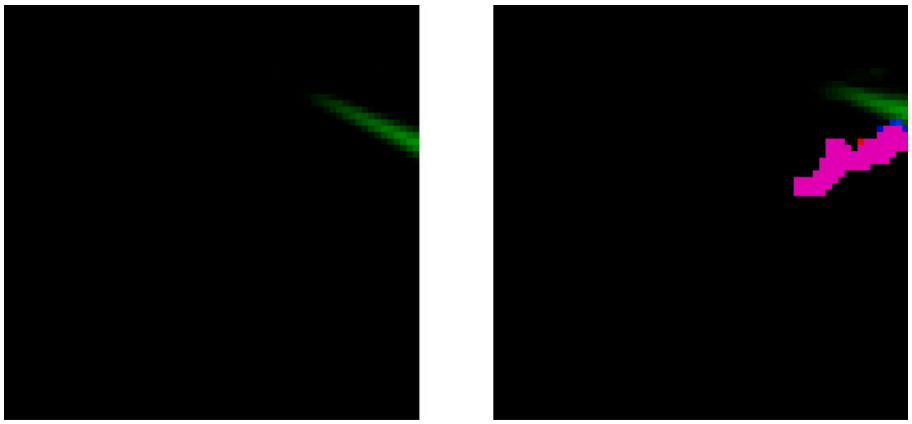
RL is very computationally expensive. One reason for this is the exploration that an agent needs to perform in the environment. To simplify solving the above MDP and to reduce the exploration space, we have used the following methods: action masking and the specification of a focus region.

4.2.1. Action masking

A common way to reduce the exploration space in RL is the use of action masking.

Our actions so far consist of separate push-down motions for each time step. However, as shown in Section 6, our real-world robot has a roller actuator, so a more continuous trajectory is actually more compatible with our robot. For this reason, we have used the following action masking approach to get a more continuous trajectory from our agent: we only allow vertices to be chosen as actions that lie immediately next to the current stencil and have not been draped yet. While the resulting stencil areas do overlap, the actual action chosen by the agent (i.e., the center point of the stencil) can never be chosen from points that have been part of a stencil before. This forces the agent to take a continuous path until there are no more neighboring vertices available. In that case, it is allowed to choose another vertex from anywhere in the region that we currently work with. This reduces the action space from size 4225 per time step to about 16 for a stencil of size 3×3, which is the size our stencil typically has for the given mesh resolution. See Fig. 8 for a graphical depiction.

The reason why action masking is employed in most RL applications is that it drastically reduces the effort necessary for exploration. Imagine we had a horizon of only 10, i.e., after 10 steps, the environment would reset. If we have 4225 possible actions per step, this amounts to $4225^{10} = 1.8 * 10^{36}$ possible trajectories (if the agent is not restricted in any way, i.e., if it can choose the same action again and again). This is practically impossible to explore, which makes action masking a necessary ingredient for practical RL applications.



(a) Initial state representation (b) State representation after 24 steps

Fig. 7. Example for an (a) initial state representation and a (b) state representation after 24 steps, showing how the wrinkle is pushed upwards by the stencil. The first channel (red) contains information about draped vertices, the second channel (green) contains the distance of the fiber material to the tool, the third channel (blue) shows the trajectory that was taken by the stencil so far. The purple path indicates a superposition of the first and third channel (red + blue), while the second channel is zero because the distance to the tool is zero. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

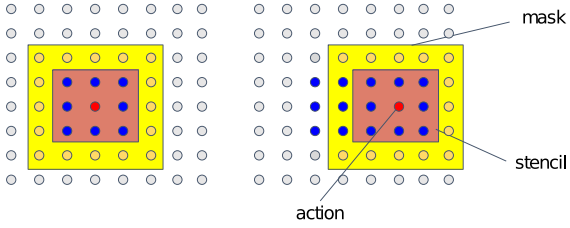


Fig. 8. Action masking for our discrete action space. The red vertex is the chosen action of the agent, the blue vertices are all the vertices that are taken to lie under the stencil and are hence pressed down. The yellow vertices depict the possible next actions. Every action that is not a direct neighbor of a stencil vertex is masked out. In addition, all vertices that have been draped are already masked out. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2.2. Focus regions

As already touched upon in Introduction to Section 4, it was not possible to tackle the problem on the whole geometry with our hardware and time constraints. That is why, besides using action masking to reduce the exploration space, we have also partitioned our problem into two sub-problems. The initial conditions that we worked with showed two wrinkles, that are relatively far apart from each other. This enabled us to focus on one wrinkle at a time by limiting the agent to only see (and being able to interact with) a sub-region of the whole setup. This way, we were able to focus our experiments on the MDP formulation, RL algorithms and hyperparameters instead of approaching a larger and potentially too difficult problem. The focus region for our test case is visible in Fig. 13(b) as the region in which draping (visible as blue vertices) has been performed.

4.3. Reinforcement learning experiments

The RL training was performed with the Proximal Policy Optimization (PPO) algorithm [46], as it is implemented in the Maze framework [47]. Additional experiments with other RL algorithms were also performed, but PPO turned out to be sufficient for this problem. The used hyperparameters can be found in Table 1.

The policy and value networks consisted of 2D convolutions with kernel size 3×3, stride 1, padding 1 and subsequent max pooling, two dense layers with 256 hidden units each and a softmax layer.

Table 1
The used PPO hyperparameters for our RL training.

Hyperparameter	Value
Learning rate	$2.5 \cdot 10^{-4}$
Discount factor	0.999
GAE λ	1.0
Batch size	400
Policy loss coeff	1.0
Value loss coeff	0.5
Entropy coeff	$2.5 \cdot 10^{-4}$
Clip range	0.2
Optimization epochs	4

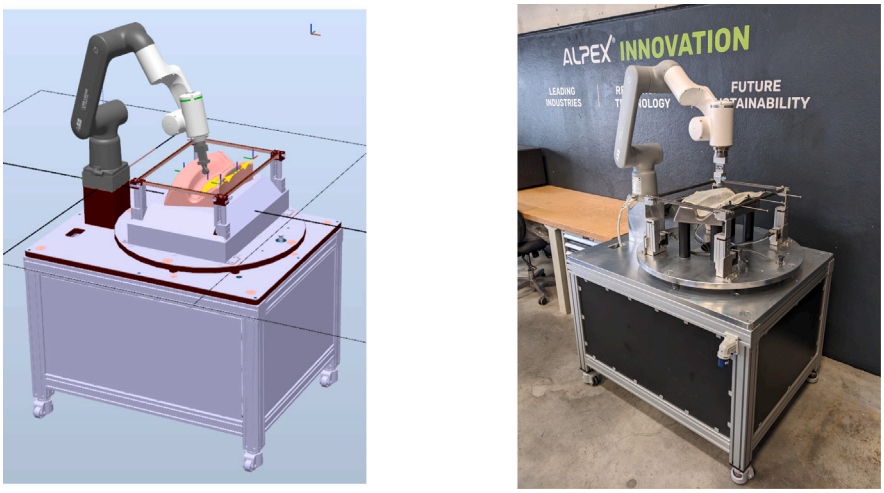
5. Comparison of modeling frameworks

The FE simulations were calculated parallelized on 4 CPUs of an Intel Core i7-8665U (1.90 GHz/2.11 GHz) processor. The baseline was calculated with the model described in Section 2.2. We tried to accelerate the FE simulation in order to check the suitability of the FE simulation model for the RL agent. Thereby, we approximated the baseline solution using mass scaling and by reducing the output frequency. The mesh density for the baseline solution was already quite coarse, thus, a further increase in mesh size did not seem feasible. To assess the time efficiency of the simulations, we compared the runtime for the FE baseline model, the accelerated FE model and the surrogate model.

To validate the simulation of the surrogate model, we transferred the optimized path suggested by the RL agent, to the FE simulation model using the VDLOAD subroutine. The draping results of both modeling systems can be exported as point cloud within an .stl file. We compared the resulting lamina geometries using the open-source software Meshlab.

6. Real-world execution environment

For the validation of the AI-proposed draping strategy, we built a physical execution environment (see Fig. 9(b)). This environment consists of an ABB CRB 15 000 Cobot and the rib tool mounted on a rotatable plate to increase the tool accessibility for the Cobot. To enable a realistic draping process, a clamping frame was installed. The frame allows for an individual positioning of the blank cut. All four corners are individually height adjustable. In addition, there are various clamping positions on all four sides of the tool. The blank cut can be



(a) Robot Studio model (b) Execution environment

Fig. 9. Comparison of the RobotStudio model of the execution environment and the real execution environment.

clamped by applying springs, fixed rods or small weights on a thread on the clamping frame.

As end-effector for the draping action, we mounted a small roller (width of 18 mm and a diameter of 30 mm) on the Cobot.

6.1. Boundary conditions

The execution environment offers many possibilities for adjusting the position and tension of the blank cut, to enable an optimized draping process. For the training of the RL agent, most of the variable boundary conditions were fixed in order to reduce parameters for the simulation and therefore shorten training duration. The chosen boundary conditions are a compromise of RL agent training duration and realistic draping conditions. In the simulation model, we assumed that the blank cut is fixed after being draped to the tool by the stencil. This boundary condition was realized by applying a small amount of spray adhesive to the bottom surface of the blank cut. This allows for a comparatively low friction before the blank cut is draped and a good fixation after being pressed to the tool by the end-effector. That way, we were able to reproduce the simulated behavior.

6.2. Draping strategy

Our goal was to replicate the draping strategy proposed by the RL agent in the execution environment. The RL agent suggests a sequential draping order for a list of points. We wrote a code in the commercial robot programming tool *Rapid* (ABB RobotStudio) to convert this point list into a path for the Cobot. Points lying close to each other were combined into one draping action: first, the roller is oriented so that it points in the direction of the next point in the list. Then, the distance from starting point to subsequent point is driven in force mode on the tool. Larger distances between two points are considered a jump during the draping process. Isolated single points (i.e., single points with large distances to the previous and subsequent point) are pressed to the tool with a given force and without movement. That way, we can reproduce the behavior of the surrogate model.

7. Results

7.1. Material

7.1.1. Material properties

Input data for the *FABRIC material model was generated by fitting the experimental data. For modeling the tensile behavior, we used a bi-linear fit for both directions. Since for the dry fabric even small compressive loads lead to buckling, there is no established method to test the compressive stiffness experimentally. Therefore, as proposed by [37], we determined the input parameters for compression phenomenologically by matching the experimentally determined bending length with the cantilever bending simulation results. For the in-plane shear behavior, only the small strain region (accounting for strains up to 10 %) was modeled using a linear fit. The transverse shear stiffness values K11 (warp direction) and K22 (weft direction) were derived from the in-plane shear data as described in [35]. We determined the friction coefficients by calculating the mean value of the experiments for the static and dynamic friction. An overview of the input parameters is shown in Table 2.

7.1.2. Material validation

The diaphragm forming simulation with the characterized material was validated by comparing the draping simulation result with 11 experimental draping results. An exemplary comparison with one of the experiments is shown in Fig. 10.

Topology. The simulation result (Fig. 10a) was compared to the experimental results (representative experimental result in Fig. 10b) in terms of topology (Fig. 10c), boundary contour (Fig. 10d) and local fiber orientations (Fig. 10e and f). For the simulation as well as the experiments, the lamina was not fully draped onto the tool but showed some bridging at the transition of hemisphere and baseplate. All 11 experimental results showed similar draping results. Concerning the topology, the experiments tend to show highest deviations at the bridging area, with an average maximum deviation of about 2.7 mm (see Table 3). For the experiments we removed the blankholders after draping (which is not the case for the simulation), resulting in negative deviations at the areas of the blankholders. Excluding those blankholder areas from the evaluation, there are only slight deviations at the bridging areas. Apart from that, simulation and experiments are in good agreement.

A common approach to evaluate the topology is to use photographs (e.g., [48–50]), only enabling a qualitative comparison between simulation and experiment. Alternatively, it is also common practice to use strain-based methods for evaluating topological differences (e.g., [51]). However, recent studies incorporate 3D evaluations of topological defects such as wrinkles and bridging. Osterberger et al. report a slight underestimation of large deviations (above 1 mm) by the simulation, which is also the case for the bridging areas in our hemisphere simulation [38].

Table 2
Material parameters used for the FE simulation model obtained by material characterization with nominal stresses σ in [MPa] and nominal strains ϵ.

Uniaxial tension								Friction			
Warp-Direction				Weft-Direction				Warp-Steel		Weft-Steel	
Tension		Compression		Tension		Compression					
σ	ϵ	σ	ϵ	σ	ϵ	σ	ϵ	μ_S	μ_D	μ_S	μ_D
0	0	0	0	0	0	0	0	0.28	0.25	0.27	0.24
3	0.005	0.12	0.02	5	0.006	0.0405	0.02				
40	0.02			60	0.025						
In-Plane shear								Bending stiffness			
Shear +45°				Shear -45°				K11	K22	K12	
σ		ϵ		σ		ϵ		27.2	27.2	0	
0		0		0		0					
0.005		0.1		0.0036		0.1					

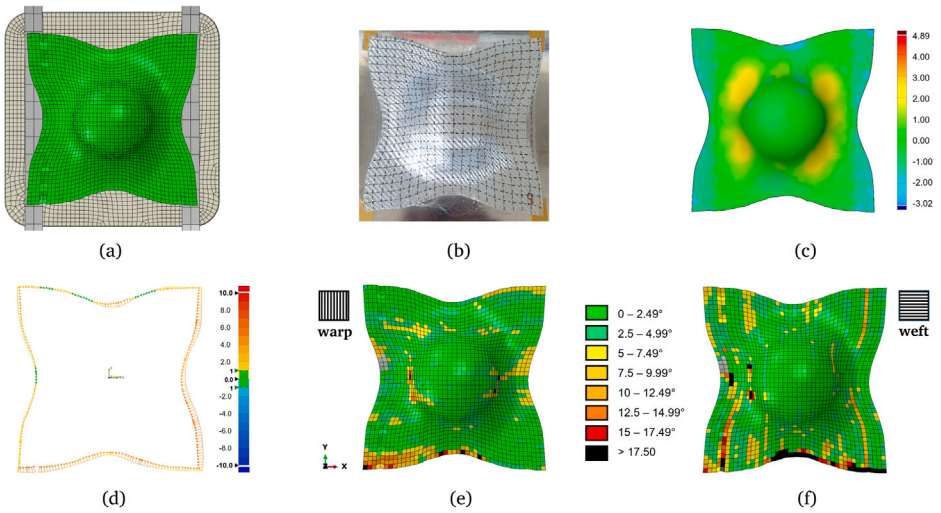


Fig. 10. Exemplary comparison of the (a) simulation with a (b) experiment result in terms of (c) topology, (d) boundary contour and fiber orientations in (e) warp and (f) weft direction.

Boundary contour. On average, the boundary contours maximally deviate by about 6.4 mm from the boundary of the simulation (with the worst result showing deviations by 8.7 mm). These deviations are mostly due to the fringing of the lamina at the boundary in the experiments which is not modeled within the FE simulation. In recent literature, evaluations on edge contours are mostly done in 2D. The presented contour deviations (c.f. Table 3) align well with values found in literature. Dörr et al. [52] investigated the draping behavior of different commercial FE codes for thermoplastic UD materials in a benchmark study, reporting deviations between 8 and 10 mm. Osterberger et al. [38] conducted a contour comparison between simulations and experiments using UD epoxy prepregs, describing deviations in a similar range. For woven fabrics (epoxy prepregs), Chen et al. [53] report deviations of about 5 mm for a hemispheric and 8 mm for pyramid-shaped tool.

Fiber angles. Overall, about 88% (warp) and 78% (weft) of fiber angles deviate by less than 5° from the simulation result. The fiber angles in weft direction tend to deviate more than in warp direction. This is explained by the blankholders which have a higher influence on the fiber orientation perpendicular to the transition line of blankholder and tool. Additionally, higher deviations are near the border of the lamina, caused by fringing at the boundary. Increased deviations at the hemispherical area are measurement artifacts of the FScan measurement because those areas were most challenging to reach for the robot.

The comparison of fiber angle deviations with literature poses a challenge because most studies either focus on shear angle evaluations or merely conduct local evaluations of fiber angles. Full-field fiber angle comparisons for UD prepregs presented in [54] reported about 95% within 4° deviation between simulation and experiment (only considering areas free from topological defects). Chen et al. evaluated global deviations in biaxial non-crimp fabrics (NCFs) within a range of 0–5° [51]. The presented validation shows slightly less accordance, mainly originating from the fringing boundaries, the blankholder areas as well as the measuring artifacts. Excluding the affected elements would result in accuracies comparable to the literature. Bai et al. evaluated fiber angles in woven fabrics locally, reporting up to 3.3° deviations [55]. This correlates well with our evaluation, where the majority of fiber angle deviations lies within 0° and 5° (see Fig. 10e and f).

7.2. Comparison FE and surrogate model

7.2.1. Results comparison with baseline

Fig. 11 shows a contour plot displaying topological deviations on the top surface of the surrogate model result. Within the masked area (red box), there are hardly any deviations to the baseline obtained by the FE model. Highest deviations occur at areas outside the tool surface area (i.e., where the lamina protrudes beyond the tool).

For the accelerated FE simulation we applied mass scaling, reduced the output frequency to a minimum and spared damping parameters. Thereby, the calculation time was reduced to 40.2 min. Fig. 12 shows the comparison of the accelerated FE model result with the baseline. As for the surrogate model result, the masked area shows no deviations. However, due to the increased mass and the lack of material damping (which also increases calculation time), distinct effects of oscillations

Table 3
Deviations of topology, boundary contour and fiber orientations between simulation and experiments.

Sample	Deviations of...					
	Topology [mm]		Boundary [mm]		Fibers < 5° [%]	
	Min	Max	Mean	Max	Warp	Weft
1	−3.0	2.3	1.8	5.4	86.8	73.8
2	−2.6	2.7	2.8	8.7	89.1	78.3
3	−1.6	2.3	2.0	5.5	88.3	78.1
4	−3.4	3.3	2.6	6.4	90.0	77.7
5	−3.9	3.3	2.5	6.3	87.9	77.5
6	−2.8	2.2	3.0	6.7	86.4	78.9
7	−2.9	3.0	2.8	6.8	85.7	82.7
8	−3.4	2.6	2.3	5.6	86.5	78.9
9	−2.7	2.6	2.6	6.0	88.5	76.4
10	−2.6	2.4	2.4	5.8	87.6	76.7
11	−3.2	2.6	2.7	6.8	90.2	80.1
Mean	**−2.9**	**2.7**	**2.5**	**6.4**	**87.9**	**78.1**

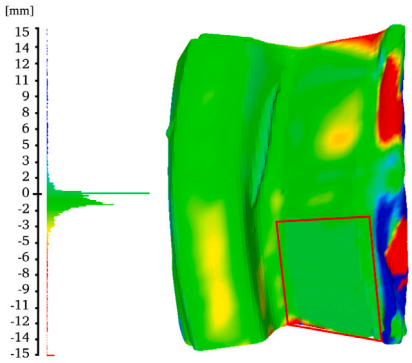


Fig. 11. Topology comparison of the simulation result from the surrogate model and the baseline solution from the FE model after smoothing the wrinkle using the path suggested by the surrogate model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

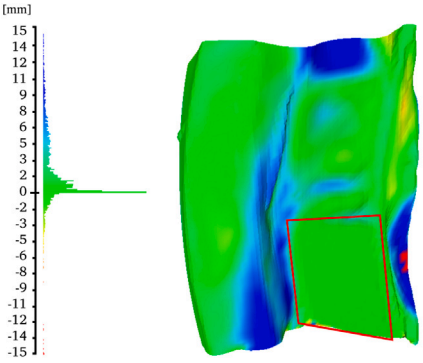


Fig. 12. Topology comparison of the simulation result from the accelerated FE model and the baseline model after smoothing the wrinkle using the path suggested by the surrogate model.

within the material are visible, leading to higher deviations to the baseline in the residual areas. Considering the deviations to the baseline and that 40.2 min is far too slow for the RL agent to determine a viable solution, the approximation using a GPU based surrogate model is inevitable.

7.2.2. Runtime comparison with FEM

The calculation time for the baseline solution using the FE model was about 8.15 h. This calculation time could be reduced to 40.2 min using classical acceleration methods for dynamic FE simulations. Despite the 92% decrease in calculation time, the accelerated simulation was still too slow for the RL agent to determine the optimum path. In comparison, a rollout of the draping strategy on the first wrinkle takes about 19 s with the surrogate simulation. This means that the surrogate model allows for runtimes reduced by three orders of magnitude, enabling suitable starting conditions for the RL agent.

7.3. RL results

The RL agent managed to successfully and fully autonomously find a trajectory for the focus region that almost completely drapes the lamina to the tool and thus removes the right-hand side wrinkle. Fig. 13 shows the states of the surrogate simulation before and after the draping. There are still minor issues to be seen in the final state: for instance, there are 6 undraped inner vertices (visible as inner black vertices in Fig. 13(b)) and there is self-penetration visible in the upper-right corner of the focus area. The undraped inner vertices point to the fact that there is still a small distance between these vertices and the tool, which means that the region has not been draped perfectly. The self-penetration is due to the fact that self-collision is not implemented for this simulation yet. These are both issues that will be addressed in future work.

Intermediate trajectory states are shown together with the real-robot trajectory in the Appendix A in Fig. A.16.

The path can also be seen in the video from the supplementary material. One sees that the RL agent is compelled to start draping the wrinkle to the tool right from the start because this is what our reward formulation encourages. Once the wrinkle is draped, the movement of the stencil becomes more or less random because the agent was only rewarded based on the sum of distance of all vertices to the tool surface. When this distance does not change any more, the reward does not increase further and learning stops. This results in the basically random jumping-around behavior that is observable in the end. One way to address this could be to experiment with different reward formulations that supply penalties for jumping too far, for instance.

A typical training run for the focus area takes about 1.5 days on two NVIDIA RTX 4090 GPU, of which one is used for the surrogate simulation and one for RL. Fig. 14 shows a reward curve of such a training run for the draping of the wrinkle within the focus area.

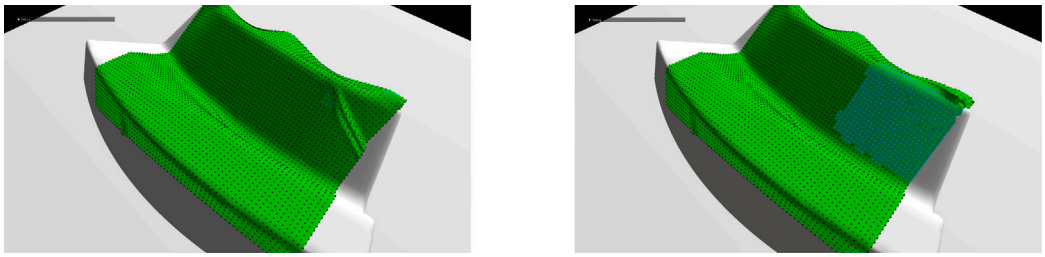
7.4. Real-world results

Fig. A.16 shows a comparison of the surrogate and the experimental draping result at different stages within the draping path. The predicted intermediate results as well as the end result match quite well with the experiment. Our goal for the execution environment was to implement an automated routine that imports the point list provided by the RL agent, assigns corresponding robot configurations, classifies the points into the different categories for the draping strategy and starts the draping process. We have realized the automation of the draping process as described in Section 6. However, we had to add manual steps in order to enable an error free robot path. We faced two challenges, preventing a fully automated routine:

Singularities. Some points of the trajectory suggested by the RL agent resulted in singularities, which led to an error. For those points, we slightly modified the inclination of the end-effector manually. In case that the singularity could not be resolved, affected points were omitted.

Collisions. Due to the complex tool geometry, some movements (automatically generated by RobotStudio using the trajectory) would have led to collisions. In those cases, we had to manually teach the robot alternative alignments of the end-effector that avoided collision.

Although the full automation of singularity and collision prevention (either in terms of continuous checks during execution or through iterative path adaption) is theoretically possible, it would lead to a dramatic increase in computational demand, contradicting the goal of



(a) Initial state of the environment. (b) Final state of the environment.

Fig. 13. Initial and final state of the environment after rolling out the optimized trajectory the RL agent determined during training. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

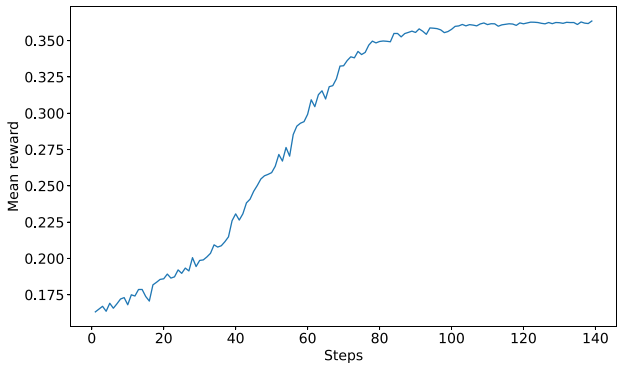


Fig. 14. Average reward over the course of an experiment to drape the first wrinkle. The y-axis shows the obtained mean reward per episode, the x-axis represents the number of PPO training steps.

our approach. The manual effort needed to readjust the robot configuration and movement fully depends on the path suggested by the agent and the position of the robot. We realized the readjustment applying standard techniques for singularity and collision avoidance. The main problem is that activated force control leads to small deviations from the planned path. This leads to errors during runtime that are not detected a priori by RobotStudio. This is mainly the case for singularities, while collisions are rarely affected by these slight deviations from the path. A potential solution to avoid singularities and collisions is to add the rotation of the tool relative to the robot as an additional degree of freedom. If singularities or collisions are detected by RobotStudio, for certain movements, the tool can be rotated stepwise for 45° or 90° until a valid solution for these movements is found. In addition to the additional degree of freedom, also a different position of the Cobot relative to the tool can help to solve the mentioned problems. Modifications of the roller design can also reduce the errors during path planning. This applies only for errors detected by RobotStudio. The errors occurring only during runtime because of the small deviations from the planned path cannot be detected in advance because the robot arm itself is not simulated. Therefore, this can also not be mitigated before a first test run. Considering the robot position and the kinematics of the robot during the training of the agent by simulating the arm explicitly could help to reduce or even avoid singularities and collisions, while increasing the computing time and limiting the agent to this predefined setting.

8. Conclusion

The glass fiber woven fabric material was successfully characterized using standard testing methods. Comparisons of experimental and numerical results of a diaphragm draping process are in good agreement and occurring deviations of topology, boundary contour as well as fiber deviations align well with values found in literature. Therefore, the use of FE simulation results as baseline solutions is plausible. We attempted to accelerate the FE simulation model using conventional acceleration methods, achieving a 92 % decrease in calculation time. Since the reduction was not enough for a RL agent to provide a feasible solution, we created a PBD surrogate model enabling interactive simulations (speedup by three orders of magnitude). Using the surrogate model, we successfully trained the RL agent to determine the optimum draping path for smoothing out the wrinkle present at the focus region within the test case. The determined draping path was transferred back to the FE simulation model in order to get a baseline solution. Within the focus region, comparisons between the resulting topologies of FE and surrogate simulation after draping were in good agreement. Compared to the accelerated FE model, the PBD surrogate achieved results more similar to the baseline simulation — additionally providing enough calculation speed to enable the application of RL.

The determined path was then transferred to the real-world execution environment. Despite difficulties in reproducing the exact wrinkle of the initial test case, the draping path proposed by the RL agent led to a smoothing out of the wrinkle. A visual comparison of the resulting geometries after path draping shows high similarities between the surrogate and experimental result.

We therefore conclude that the application of RL for the draping problem of woven fabrics is actually a promising direction if one can replace the computationally expensive FE model by a cheaper surrogate simulation. This study was meant as a proof of concept to demonstrate the basic feasibility.

9. Future work

The promising results obtained in our study, demonstrating the feasibility of training a draping agent in simulation within a reasonable time frame and successfully applying its output to a real robot, open several avenues for future research. One significant opportunity lies in creating a fully automated system for draping cloth materials over any tool geometries using collaborative robots. This can be achieved by integrating cutting-edge technologies such as 3D scanning, cloth simulation, reinforcement learning, and robotic control, as depicted in Fig. 15.

1. **Real-World Setup Scanning:** Use high-resolution 3D scanning technologies, such as structured light or laser scanning systems [38,56], to accurately capture the 3D geometry of the tool and the initial state of the cloth. These systems would generate detailed 3D models that serve as the foundational data for subsequent simulation and planning stages.
2. **Cloth Draping Simulation:** The 3D models obtained from the scanning process would be transferred into a cloth draping simulator. This simulator would need to accurately replicate the physical interactions between the cloth and the tool. To ensure computational efficiency, surrogate models designed for fast

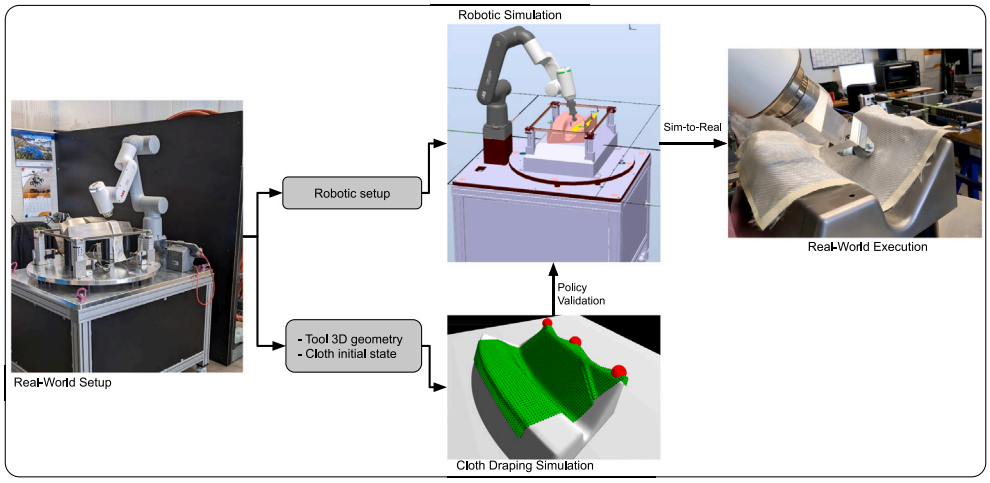


Fig. 15. Development pipeline for fully automated draping. The real-world setup, featuring the robot with the cloth to be draped on a given tool, informs both the robotic simulation setup and the cloth draping simulation. The draping simulation trains a policy, which is validated in the robotic simulation before being transferred to the real robot for execution.

cloth draping simulation as the one presented in this work would be employed. These models would allow for rapid iteration and evaluation of different draping strategies.

3. **Reinforcement Learning Policy Training:** With a fairly accurate and rapid simulation environment established, a RL policy can be trained to determine the optimal draping path. The RL agent would use a small stencil, representing the robot gripper, to perform the draping actions. The training process would involve learning from the surrogate model to develop a policy that effectively drapes the cloth over the tool with minimal topological defects. To facilitate the Sim-to-Real transfer, it would be beneficial to investigate varying stencil sizes and different methods of applying the stencil to the cloth. Additionally, Imitation Learning can be employed by having the RL agent observe and learn from an expert performing the task [57]. This can significantly enhance the initial training phase by providing the agent with a set of optimal behaviors to mimic. Combining reinforcement learning with imitation learning enables the agent to leverage expert demonstrations to improve performance and reduce training time.
4. **Robotic Simulation and Path Validation:** The draping path generated by the RL agent would be transferred to a robotic simulator. This simulator would utilize an inverse dynamics solver to attempt to reproduce the same path. Validating the robotic actions in this simulated environment ensures they are feasible and free from singularities or other technical issues. Any encountered issues in reproducing the provided paths would suggest modification on how the RL agent interacts with the cloth to ensure a seamless transfer to the simulated robot. Additionally, sim-to-real state-of-the-art techniques can be integrated [58].
5. **Real-World Execution:** If the robotic simulation confirms that the generated path is valid, the action sequence would be transferred to the real-world robotic system. The robot would then execute the draping task according to the learned policy. In cases where the simulated path encounters issues, such as singularities or a discrepancy in the outcome of the executed actions, the automation system would adjust the action space in the cloth simulation and modify relevant parameters.
6. **Continuous Improvement and Adaptation:** Continuous improvement of the surrogate model and RL policy would be achieved through feedback from both simulations and real-world executions. This iterative improvement loop would refine the system's performance and its capability to adapt to different types of cloth materials, initial states and tool geometries.

Building a fully automated system would lead to more precise and efficient fabric draping, paving the way for innovative automated manufacturing processes.

CRediT authorship contribution statement

Patrick M. Blies: Writing – review & editing, Writing – original draft, Visualization, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Sophia Keller:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis. **Ulrich Kuenzer:** Writing – review & editing, Writing – original draft, Software, Project administration, Investigation. **Yassine El Manyari:** Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis. **Franz Maier:** Writing – review & editing, Funding acquisition, Conceptualization. **Markus G.R. Sause:** Writing – review & editing, Supervision. **Marcel Wasserer:** Software, Project administration, Funding acquisition. **Roland M. Hinterhölzl:** Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ulrich Kuenzer reports financial support was provided by Republic of Austria Federal Ministry for Climate Action Environment Energy Mobility Innovation and Technology. If there are other authors, they 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

The authors want to thank the Austrian Research Promotion Agency (FFG) for the financial support of the project Jarvis4Pre (Grant No. 886871) in the scope of the "Take Off" Funding Program of the Republic of Austria, Ministry for Climate Action.

Appendix A. Comparison of RL and robot trajectories

See Fig. A.16.

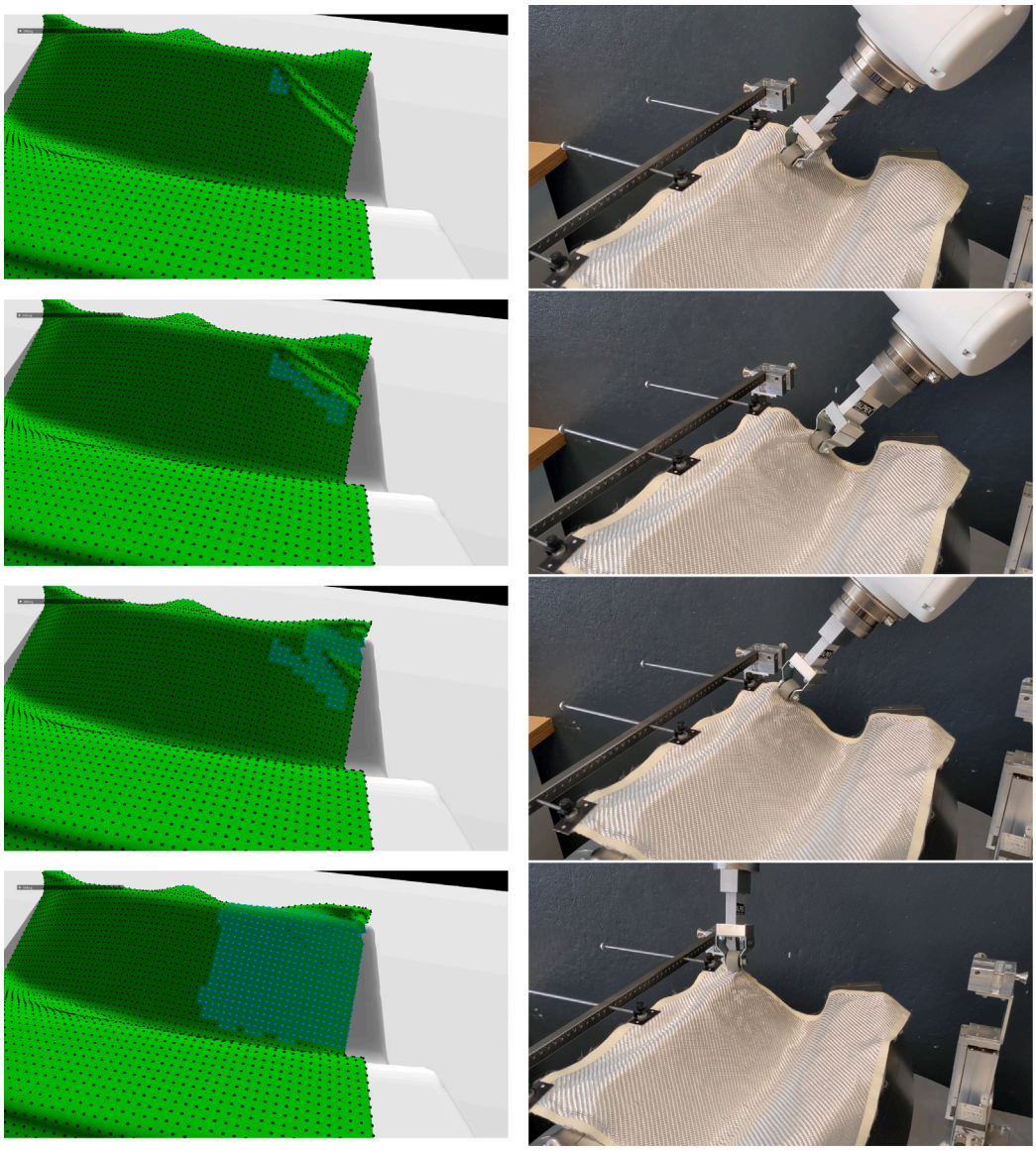


Fig. A.16. Snapshots of the optimized trajectory that was obtained by the RL agent and its playback on the real-world robot.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jmapro.2025.02.063.

References

[1] Breitschopf C. 2024, https://www.drapebot.eu/.

[2] Manohar MD. Module 6-applications of composites introduction (1). 2014, URL: http://www.compositesworld.com/articles/the-outlook-for-thermoplastics-in-aerospace-2014-2023.

[3] Bloom LD. On the relationship between layup time, material properties and mould geometry in the manufacture of composite components (Ph.D. thesis), University of Bristol; 2015.

[4] Lukaszewicz DH-J, Ward C, Potter KD. The engineering aspects of automated prepreg layup: History, present and future. Compos Part B: Eng 2012;43(3):997–1009. http://dx.doi.org/10.1016/j.compositesb.2011.12.003, URL: https://www.sciencedirect.com/science/article/pii/S1359836811005452.

[5] Kako J, Roth Y. Applications and challenges of prepreg forming technologies in aircraft industry. 2015.

[6] Trinh M, Dammers H, Behery M, Baier R, Henn T, Gossen D, Corves B, Kowalewski S, Nitsch V, Lakemeyer G, Gries T, Brecher C. Safety of human-robot collaboration within the internet of production. 2023, p. 86–103. http://dx.doi.org/10.1007/978-3-031-36049-7_7.

[7] Eitzinger C, Frommel C, Ghidoni S, Villagrossi E. System concept for human-robot collaborative draping. 2021.

[8] Malhan RK, Shembekar AV, Kabir AM, Bhatt PM, Shah B, Zanio S, Nutt S, Gupta SK. Automated planning for robotic layup of composite prepreg. Robot Comput-Integr Manuf 2021;67:102020. http://dx.doi.org/10.1016/j.rcim.2020.102020, URL: https://www.sciencedirect.com/science/article/pii/S0736584520302313.

[9] Patil S, Vasu V, Srinadh KVS. Advances and perspectives in collaborative robotics: a review of key technologies and emerging trends. Discov Mech Eng 2023;2. http://dx.doi.org/10.1007/s44245-023-00021-8.

[10] Chen Y-W, Joseph RJ, Kanyuck A, Khan S, Malhan RK, Manyar OM, McNulty Z, Wang B, Barbič J, Gupta SK. A Digital Twin for Automated Layup of Prepreg Composite Sheets. J Manuf Sci Eng 2021;144(4):041010. http://dx.doi.org/10.1115/1.4052132.

[11] Pfrommer J, Zimmerling C, Liu J, Kärger L, Henning F, Beyerer J. Optimisation of manufacturing process parameters using deep neural networks as surrogate models. 72, Elsevier B.V.; 2018, p. 426–31. http://dx.doi.org/10.1016/j.procir.2018.03.046,

[12] Jakumeit J, Herdy M, Nitsche M. Parameter optimization of the sheet metal forming process using an iterative parallel kriging algorithm. Struct Multidiscip Optim 2005;29:498–507. http://dx.doi.org/10.1007/s00158-004-0455-3.

[13] Bonte MHA, Boogaard AHVD, Huétink J. A metamodel based optimisation algorithm for metal forming processes. 2007.

[14] Pfaff T, Fortunato M, Sanchez-Gonzalez A, Battaglia PW. Learning mesh-based simulation with graph networks. In: 9th international conference on learning representations, ICLR 2021, virtual event, Austria, May 3-7, 2021. OpenReview.net; 2021, URL: https://openreview.net/forum?id=roNqYL0_XP.

[15] Zimmerling C, Dörr D, Henning F, Kärger L. A machine learning assisted approach for textile formability assessment and design improvement of composite components. Compos Part A: Appl Sci Manuf 2019;124. http://dx.doi.org/10.1016/j.compositesa.2019.05.027.

[16] Zimmerling C, Poppe C, Kärger L. Estimating optimum process parameters in textile draping of variable part geometries - a reinforcement learning approach. 47, Elsevier B.V.; 2020, p. 847–54. http://dx.doi.org/10.1016/j.promfg.2020.04.263,

[17] Keller S, Maier F, Blies PM, Hinterhölzl RM. Creating training data for surrogate models using FE draping simulation. In: Proceedings of the 23rd international conference on composite materials. ICCM23, Belfast; 2023.

[18] Würth T, Freymuth N, Zimmerling C, Neumann G, Kärger L. Physics-informed MeshGraphNets (PI-MGNs): Neural finite element solvers for non-stationary and nonlinear simulations on arbitrary meshes. 2024, http://dx.doi.org/10.48550/ARXIV.2402.10681, CoRR abs/2402.10681 arXiv:2402.10681.

[19] Terzopoulos D, Platt JC, Barr AH, Fleischer KW. Elastically deformable models. In: Stone MC, editor. Proceedings of the 14th annual conference on computer graphics and interactive techniques, SIGGRAPH 1987, anaheim, california, USA, July 27-31, 1987. ACM; 1987, p. 205–14. http://dx.doi.org/10.1145/37401.37427.

[20] Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. J Vis Commun Image Represent 2007;18(2):109–18. http://dx.doi.org/10.1016/J.JVCIR.2007.01.005.

[21] Stuyck T, Chen H. DiffXPBD: Differentiable position-based simulation of compliant constraint dynamics. Proc ACM Comput Graph Interact Tech 2023;6(3):51:1–51:14. http://dx.doi.org/10.1145/3606923.

[22] Weiss T. Fast position-based multi-agent group dynamics. Proc ACM Comput Graph Interact Tech 2023;6(1):14:1–14:15. http://dx.doi.org/10.1145/3585507.

[23] Morais LZ, Bergmann VK, Carvalho EA, Zimmer R, Martins MG, Nedel LP, Maciel A, Torchelsen RP. An enhanced interactive endoscope model based on position-based dynamics and cosserat rods for colonoscopy simulation. Comput Graph 2023;116:345–53. http://dx.doi.org/10.1016/J.CAG.2023.08.020.

[24] Demirel D, Smith J, Kockara S, Halic T. GPU based position based dynamics for surgical simulators. In: Fang X, editor. HCI in games - 5th international conference, HCI-games 2023, held as part of the 25th HCI international conference, HCII 2023, copenhagen, Denmark, July 23-28, 2023, proceedings, part i. Lecture notes in computer science, vol. 14046, Springer; 2023, p. 81–8. http://dx.doi.org/10.1007/978-3-031-35930-9_6.

[25] Segato A, Di Vece C, Zucchelli S, Di Marzo M, Wendler T, Azampour MF, Galvan S, Secoli R, Momi ED. Position-based dynamics simulator of brain deformations for path planning and intra-operative control in keyhole neurosurgery. IEEE Robot Autom Lett 2021;6(3):6061–7. http://dx.doi.org/10.1109/LRA.2021.3090016.

[26] Peng H, Li N, Jiang D, Li F. Soft robot fast simulation via reduced order extended position based dynamics. Robot Auton Syst 2024;175:104650. http://dx.doi.org/10.1016/J.ROBOT.2024.104650.

[27] Liu F, Su E, Lu J, Li M, Yip MC. Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics. IEEE Robot Autom Lett 2023;8(7):3964–71. http://dx.doi.org/10.1109/LRA.2023.3264766.

[28] Ruiz JCS, Mula J, Poler R. Job shop smart manufacturing scheduling by deep reinforcement learning. J Ind Inf Integr 2024;38:100582. http://dx.doi.org/10.1016/J.JII.2024.100582.

[29] Lu F, Zhou G, Zhang C, Liu Y, Chang F, Xiao Z. Energy-efficient multi-pass cutting parameters optimisation for aviation parts in flank milling with deep reinforcement learning. Robot Comput Integr Manuf 2023;81:102488. http://dx.doi.org/10.1016/J.RCIM.2022.102488.

[30] Hu C, Saboo KV, Ali AH, Juran BD, Lazaridis KN, Iyer RK. REMEDI: reinforcement learning-driven adaptive metabolism modeling of primary sclerosing cholangitis disease progression. In: Hegselmann S, Parziale A, Shanmugam D, Tang S, Asiedu MN, Chang S, Hartvigsen T, Singh H, editors. Machine learning for health, ML4H@neurIPS 2023, 10 December 2023, new orleans, louisiana, USA. Proceedings of machine learning research, vol. 225, PMLR; 2023, p. 157–89, URL: https://proceedings.mlr.press/v225/hu23a.html.

[31] Meier D, Ensari I, Konigorski S. Designing and evaluating an online reinforcement learning agent for physical exercise recommendations in N-of-1 trials. In: Hegselmann S, Parziale A, Shanmugam D, Tang S, Asiedu MN, Chang S, Hartvigsen T, Singh H, editors. Machine learning for health, ML4H@neurIPS 2023, 10 December 2023, new orleans, louisiana, USA. Proceedings of machine learning research, vol. 225, PMLR; 2023, p. 340–52, URL: https://proceedings.mlr.press/v225/meier23a.html.

[32] Luo J, Dong P, Wu J, Kumar A, Geng X, Levine S. Action-quantized offline reinforcement learning for robotic skill learning. In: Tan J, Toussaint M, Darvish K, editors. Conference on robot learning, coRL 2023, 6-9 November 2023, atlanta, GA, USA. Proceedings of machine learning research, 229, PMLR; 2023, p. 1348–61, URL: https://proceedings.mlr.press/v229/luo23a.html.

[33] Alkhoury G, Berri S, Chorti A. Deep reinforcement learning-based network slicing algorithm for 5g heterogenous services. In: IEEE global communications conference, GLOBECOM 2023, kuala lumpur, Malaysia, December 4-8, 2023. IEEE; 2023, p. 5190–5. http://dx.doi.org/10.1109/GLOBECOM54140.2023.10436893.

[34] Di Cao, Zhao J, Hu J, Pei Y, Huang Q, Chen Z, Hu W. Physics-informed graphical representation-enabled deep reinforcement learning for robust distribution system voltage control. IEEE Trans Smart Grid 2024;15(1):233–46. http://dx.doi.org/10.1109/TSG.2023.3267069.

[35] Dassault Systèmes. Abaqus 2021 theory manual. Providence, RI, USA: Dassault Systèmes Simulia Corp.; 2021, https://abaqus-docs.mit.edu/2017/English/SIMACAEANLRefMap/simaanl-c-expdynamic.htm.

[36] Pickett AK. Process and mechanical modelling of engineering composites. Stuttgart: University of Stuttgart; 2018.

[37] Osterberger J, Maier F, Hinterhölzl RM. Application of the abaqus *fabric model to approximate the draping behavior of UD prepregs based on suited mechanical characterization. Front Mater 2022;9. http://dx.doi.org/10.3389/fmats.2022.865477.

[38] Osterberger J, Maier F, Keller S, Hinterhoelzl R. Evaluation of draping simulations by means of 3D laser scans and robot supported fiber angle scans. Front Mater 2023;10. http://dx.doi.org/10.3389/fmats.2023.1133788.

[39] Carvalho AA. Tips and tricks for explicit simulations. 2019.

[40] Macklin M, Müller M, Chentanez N. XPBD: position-based simulation of compliant constrained dynamics. In: Neff M, Geraerts R, Shum HPH, editors. Proceedings of the 9th international conference on motion in games, MIG 2016, burlingame, california, USA, October 10-12, 2016. ACM; 2016, p. 49–54. http://dx.doi.org/10.1145/2994258.2994272.

[41] Bender J, Müller M, Macklin M. A survey on position based dynamics. In: Bousseau A, Gutierrez D, editors. 38th annual conference of the European association for computer graphics, eurographics 2017 - tutorials, lyon, France, April 24-28, 2017. Eurographics Association; 2017, http://dx.doi.org/10.2312/EGT.20171034.

[42] Hu Y, Li T, Anderson L, Ragan-Kelley J, Durand F. Taichi: a language for high-performance computation on spatially sparse data structures. ACM Trans Graph 2019;38(6):201:1–201:16. http://dx.doi.org/10.1145/3355089.3356506.

[43] Macklin M, Storey K, Lu M, Terdiman P, Chentanez N, Jeschke S, Müller M. Small steps in physics simulation. In: Lee S, Schroeder CA, Spencer SN, Batty C, Huang J, editors. Proceedings of the 18th annual ACM SIGGRAPH/eurographics symposium on computer animation, SCA 2019, los angeles, CA, USA, July 26-28, 2019. ACM; 2019, p. 2:1–7. http://dx.doi.org/10.1145/3309486.3340247.

[44] Bellman R. A Markovian decision process. Indiana Univ Math J 1957;6:679–84.

[45] LeCun Y, Boser BE, Denker JS, Henderson D, Howard RE, Hubbard WE, Jackel LD. Handwritten digit recognition with a back-propagation network. In: Touretzky DS, editor. Advances in neural information processing systems 2, [nIPS conference, denver, colorado, USA, November 27-30, 1989]. Morgan Kaufmann; 1989, p. 396–404, URL: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.

[46] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017, CoRR, abs/1707.06347, arXiv:1707.06347 URL: http://arxiv.org/abs/1707.06347.

[47] enliteAI. Maze – applied reinforcement learning with python. 2022, https://github.com/enlite-ai/maze.

[48] Margossian A. Forming of tailored thermoplastic composite blanks: material characterisation, simulation and validation. 2016, URL: www.lcc.mw.tum.de.

[49] Thompson AJ, Belnoue JP, Hallett SR. Modelling defect formation in textiles during the double diaphragm forming process. Compos B: Eng 2020;202. http://dx.doi.org/10.1016/j.compositesb.2020.108357.

[50] Huang J, Boisse P, Hamila N, Gnaba I, Soulat D, Wang P. Experimental and numerical analysis of textile composite draping on a square box. Influence of the weave pattern. Compos Struct 2021;267. http://dx.doi.org/10.1016/j.compstruct.2021.113844.

[51] Chen S, McGregor OP, Harper LT, Endruweit A, Warrior NA. Defect formation during preforming of a bi-axial non-crimp fabric with a pillar stitch pattern. Compos A: Appl Sci Manuf 2016;91:156–67. http://dx.doi.org/10.1016/j.compositesa.2016.09.016.

[52] Dörr D, Brymerski W, Ropers S, Leutz D, Joppich T, Kärger L, Henning F. A benchmark study of finite element codes for forming simulation of thermoplastic UD-tapes. 66, Elsevier B.V.; 2017, p. 101–6. http://dx.doi.org/10.1016/j.procir.2017.03.223,

[53] Chen B, Colmars J, Naouar N, Boisse P. A hypoelastic stress resultant shell approach for simulations of textile composite reinforcement forming. Compos A- Appl Sci Manuf 2021;149:106558, URL: https://api.semanticscholar.org/CorpusID:237722547.

[54] Keller S, Maier F, Osterberger J, Hinterhölzl RM. Comparing local fiber angles from draping experiments to simulations. In: Proceedings of the 20th European conference on composite materials. ECCM20, Lausanne. 2022.

[55] Bai R, Colmars J, Chen B, Naouar N, Boisse P. The fibrous shell approach for the simulation of composite draping with a relevant orientation of the normals. Compos Struct 2022;285. http://dx.doi.org/10.1016/j.compstruct.2022.115202.
[56] He C, Shen Y, Forbes A. Towards higher-dimensional structured light. Light Sci Appl 2022;11. URL: https://api.semanticscholar.org/CorpusID:250286360.
[57] Zheng B, Verma S, Zhou J, Tsang IW-H, Chen F. Imitation learning: Progress, taxonomies and challenges. IEEE Trans Neural Netw. Learn Syst 2021;35:6322–37, URL: https://api.semanticscholar.org/CorpusID:253080956.
[58] Zhao W, Queralta JP, Westerlund T. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. 2020 IEEE Symp Ser Comput Intell (SSCI) 2020;737–44, URL: https://api.semanticscholar.org/CorpusID:221971078.