Reinforcement learning for fuselage shape control during aircraft assembly

Tim Lutz, Yinan Wang, Xiaowei Yue & Jaime Camelio

To cite this article: Tim Lutz, Yinan Wang, Xiaowei Yue & Jaime Camelio (19 Nov 2024): Reinforcement learning for fuselage shape control during aircraft assembly, IISE Transactions, DOI: 10.1080/24725854.2024.2413136

To link to this article: https://doi.org/10.1080/24725854.2024.2413136

IISE TRANSACTIONS
https://doi.org/10.1080/24725854.2024.2413136

Reinforcement learning for fuselage shape control during aircraft assembly

Tim Lutz[a], Yinan Wang[b], Xiaowei Yue[c], and Jaime Camelio[d]

[a]Department of Management Science, Darla Moore School of Business, University of South Carolina, Columbia, SC, USA; [b]Department of Industrial and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA; [c]Department of Industrial Engineering, Tsinghua University, China; [d]The College of Engineering, The University of Georgia, Athens, GA, USA

ABSTRACT
Critical safety requirements necessitate ultra-high precision quality control during the assembly of large aerospace components to reduce the mismatch between parts to be joined. Traditional methods use heuristic shape adjustment or surrogate model-based control. These methods are limited by reliance on accurate model learning and inadequate robustness to varying initial assembly conditions. To address these limitations, this article proposes a model-free reinforcement learning approach for adaptive fuselage shape control during aircraft assembly. The trained reinforcement learning agent directly adjusts the aircraft components in response to their part variations and enables an autonomous system (like AlphaGo) to learn the optimal shape control policy. Specifically, the reinforcement learning environment is built on the finite element simulator. A reward function is developed to capture the optimization objective and introduces a scheme to enforce the original constraints. The proximal policy optimization algorithm is modified to speed up the learning progress and achieve better final performance. In the case study, the root-mean-square gap between components is reduced by 98.4% on average compared with their initial shape mismatch. Our proposed method outperforms the benchmark methods with smaller final shape errors, smaller maximum forces, and lower variations across different test samples.

ARTICLE HISTORY
Received 22 June 2023
Accepted 19 September 2024

KEYWORDS
Reinforcement learning; aircraft assembly; shape control; quality Management

1. Introduction

In recent years, composite materials such as Carbon Fiber Reinforced Polymers (CFRP) and fiberglass-reinforced plastics have become ubiquitous in commercial aerospace applications; they now comprise more than 50% in flagship passenger airplanes of the major manufacturers. These composites possess superior stiffness-to-weight ratios and corrosion resistance and can be specifically tuned for a particular application. However, producing large primary structures (e.g., fuselages, wing boxes) from composites has not been without complications. For example, while monolithic construction techniques reduce the number of fasteners that are required in the assembly of an airplane, they also require ultra-high precision for successful assembly, because there are fewer opportunities to correct geometric shape errors. Over the past few years, repeated problems with the assembly of fuselages made from CFRP have led the Federal Aviation Administration (FAA) to intervene and force deliveries from a major manufacturer to be halted (Schaper, 2022).

The shape adjustment and dimensional quality control of large composite components in aircraft assembly are very challenging, due to two major reasons: (i) complex properties of composite materials. Composite materials are highly nonlinear, anisotropic, and compliant, which have quite different mechanical properties from conventional aerospace materials such as Aluminum and Titanium alloys. Existing physical model-driven quality control approaches do not work well; (ii) ultra-high precision quality requirement. The precision of composite fuselage assembly may be as high as 0.007 inches, which raises new challenges for conventional quality management. Considering varying assembly conditions and initial deformations, such a high precision is very challenging. There have been several efforts to address the shape adjustment and quality control problem in the context of large aerospace parts, particularly for fuselage assembly, including physics-driven methods and data-driven methods. For a detailed literature review, refer to Section 2.1. Our main contribution in this article is to propose a new reinforcement learning approach for shape adjustment and quality control in aircraft assembly.

Reinforcement Learning (RL) has garnered a lot of attention in recent years for its ability to exceed human-level performance in playing games, e.g., Atari games (Mnih *et al.*, 2015), Starcraft (Vinyals *et al.*, 2019), Chess, and perhaps most notably AlphaGo and AlphaZero (Silver *et al.*, 2018). Advances in RL have also originated from efforts to apply it to classic control tasks (e.g., cartpole, mountain car, inverted pendulum) (Bertsekas, 2019), chip design (Mirhoseini *et al.*, 2021), human–robot interaction (Oliff *et al.*, 2020), and so on. RL has several advantages: (i) it can enable adaptive decision-making by interacting with environments dynamically and adjusting policies in real time; (ii) it can optimize

CONTACT Xiaowei Yue yuex@tsinghua.edu.cn
Supplemental data for this article is available online at https://doi.org/10.1080/24725854.2024.2413136

their actions based on feedback from the environment and reduce the need for manual intervention; (iii) it can handle the complexities and uncertainties in engineering systems by learning robust policies that can adapt to different situations and make reliable decisions; (iv) it leverages the knowledge learned from offline simulations and model training. Therefore, RL has the potential to improve manufacturing systems with dynamical patterns, complex environments, and inherent uncertainties and variations.

Unfortunately, the use of RL in the manufacturing space has not taken hold thus far due to unique challenges in this domain. For one, there is often a limited set of data available, and generating new data through experimentation is generally not cheap. In particular, conducting experiments on a running production line usually comes at a large cost of time and resources. Furthermore, real-world problems encountered in manufacturing are often more complex than the scenarios on which much of RL has been built. There are a few studies on using RL in manufacturing systems, such as additive manufacturing (Chung *et al.*, 2022), scheduling of semiconductor manufacturing (Park *et al.*, 2019), biological manufacturing (Ueda *et al.*, 2000), sheet metal parts assembly Wang *et al.* (2024), and other manufacturing systems (Li *et al.*, 2023).

The main contribution of this article is to develop a RL approach for fuselage shape control during aircraft assembly. As far as we know, this is the first RL exploration in the aircraft assembly field. In our study, we develop a RL environment and apply a modified Proximal Policy Optimization (PPO) algorithm to the task of shape control during fuselage assembly. Our detailed technical contributions are listed as follows:

- We formulate a compliant part shape control problem for RL. As part of this, we define a reward function that captures the objective of shape control optimization.
- We set up a finite element simulator as an RL environment and build an interface with an RL agent that enforces the associated constraints on the number and magnitude of applied forces.
- We propose a modified PPO algorithm implementation, in which we anneal the variance of the probability distribution from which actions are sampled during training. Our modification can guide the RL agent's learning procedure, accelerate convergence, and improve the performance of an RL agent whose actions are to be deterministic once deployed.

The proposed RL approach has the potential to reduce human operation errors, increase productivity, adapt to varying initial deformations of components, enhance robustness, and enable continuous quality improvement for multi-stage aircraft assembly. It provides a new pillar for the digitalization and intelligence in advanced aircraft assembly.

The remainder of this article is organized as follows: Section 2 discusses the literature review on composite structures assembly and RL in manufacturing, Section 3 proposes our RL algorithm for fuselage shape control, Section 4 discusses the method evaluation based on case study Section 5 concludes the article with a brief summary.

2. Literature review

2.1. Assembly of thin-walled composite parts in aerospace manufacturing

Primary aerospace structures rely on thin-walled panels joined to internal rib structures to achieve superior high strength-to-weight ratios. Prior to being joined as a supporting member, thin-walled panels are very compliant. As a result, joining operations tend to be difficult to control and model, in particular when non-nominal shapes are to be considered. Past works have treated process parameter optimization, error propagation, comprehensive quality control, and simulation model calibration. For example, Zhang *et al.* (2021) considered the assembly of a wing box panel to an internal skeleton and applied a genetic algorithm for multi-objective optimization to determine the ideal clamping forces in response to non-ideal parts. They relied on a Finite Element (FE) simulation and validated their results against physical experiments. Considering the assembly quality across multiple manufacturing steps, digital twin simulation approaches have been devised to measure and control quality targets via physical models (Cai *et al.*, 2021). Guo *et al.* (2023) analyzed the quantifiable and controllable assembly of thin-walled structures to avoid out-of-tolerance and deformation rebound errors. The assembly quality was improved by dynamic stiffness matrix learning, physical modeling, and inverse optimization on assembly parameters. Manohar *et al.* (2018) developed a sparse sensing and machine learning scheme to predict gaps along the wing-to-body joint on an aircraft and determined the appropriate shim size to be fabricated to fill these gaps.

Some researchers have focused on the assembly of composite fuselage sections. Wen *et al.* (2018) developed an accurately calibrated FE model that characterized the behavior of a composite fuselage in response to shape control actuator forces. Starting with this calibrated model, a number of control strategies have subsequently been devised. Yue *et al.* (2018) built a surrogate model considering uncertainties and performed a multivariate optimization to come up with a feed-forward control strategy for determining actuator forces. Du *et al.* (2019) proposed a sparse learning method that chooses a prescribed number of actuator locations from a set of candidate locations and specifies forces for shape adjustments of a single fuselage. In this work, the optimization objective was to reduce the mean gap around the joint. Following up on this work, another sparse learning method with a different objective of reducing the maximum (rather than mean) gap was introduced in Du *et al.* (2022), now considering simultaneous shape adjustments for two mating fuselage parts. Physics-constrained Bayesian optimization approaches are developed to optimize the actuators' placement and achieve superior performance (AlBahar *et al.*, 2022; Wang and Yue, 2024). Neural network Gaussian process considering input uncertainties (Lee *et al.*, 2020) and Gaussian process extension (Wang *et al.*, 2022) have been developed for predicting the dimensional deformation in shape adjustment. While the aforementioned strategies relied on statistical models derived from FE analysis for their optimizations, Zhong *et al.* (2022) exported a reduced-order FE model from the simulator and used it to solve the actuator force optimization problem. Towards utilizing online measurements in an efficient manner, Mou *et al.* (2023) considered a sparse sensor placement problem and proposed an adaptive control strategy that aims to correct for deviations between models used in optimization and physical parts seen in production.

Although these approaches achieved very good performance in the fuselage shape adjustment, they have a few limitations: (i) limited adaptability due to pre-defined model structures and assumptions; (ii) high dependence on accurate and reliable models; (iii) possible performance degradation or instability from the model errors; (iv) requirement of high-quality and sufficient data samples. RL has the potential to adapt and learn from real-time interactions between the system and the environments and overcome these limitations.

2.2. RL in manufacturing

In RL, an agent interacts with an environment. In response to a current state, it selects actions adaptively and receives a reward. Many accomplishments of RL have relied on simulated environments such as OpenAI gym (Brockman *et al.*, 2016) or arcade games (Machado *et al.*, 2018). In real-world applications, RL faces challenges that are yet to be resolved. Amongst them are limited access to samples, large continuous action spaces, and satisfying environmental constraints (Dulac-Arnold *et al.*, 2021). These challenges relate to the need for the agent to efficiently explore its environment to learn a successful policy. There exist many approaches to exploration strategies, which can be grouped into uncertainty-oriented and intrinsically motivated strategies (Hao *et al.*, 2023). Perhaps the simplest exploration strategy is the so-called ϵ-greedy strategy (Tokic, 2010). In this scheme, either the action that the agent currently rates as the best is performed (with probability $1-\epsilon$), or else a completely random action is executed. Although simple, this scheme also tends to be sample-inefficient. However, in an online setting, random actions can lead to erratic and dangerous behaviors. In off-policy algorithms, the action is often construed by adding noise to a deterministic action that is selected by the agent. The role of action noise has been identified as having an impact on the learning progress by Hollenstein *et al.* (2022), who determined that the action noise scale can make or break success in training a policy that achieves the desired goal. Accordingly, they argue that it is necessary to tune the action noise scale for the learning task at hand.

When applying RL in a real-life setting, challenges arise that tend to be of little concern in simulated or virtual environments. In the physical world, erratic actions are particularly problematic in motion control problems. Jerky motions can damage components, due to mechanical wear and excessive heat buildup. Unfortunately, superimposing Gaussian noise to actions in order to explore the environment has exactly that effect. Towards achieving smooth motions during online training with a robot manipulator, Raffin *et al.* (2022) modified state-dependent exploration as follows: instead of generating noise from the initial state of an episode, they take policy features as the input to a noise-generating function at regular intervals of n steps. As an alternative to adding noise to actions, Plappert *et al.* (2017) proposed to instead introduce noise to the agent's parameters, perturbing them at the start of a learning episode. This idea is borrowed from evolutionary methods. By doing so, the actions for a given state are deterministic, i.e., they are repeatable. This is not the case when random noise is added after the agent selects an action. These strategies can be complex and rely on the agent to automatically adjust the exploration/exploitation trade-off during training. The current article takes a different approach in which we exert direct control over the exploration/exploration scheme as learning progresses. This allows us to incorporate prior knowledge about the RL environment and to drive the agent to learn its policy according to the desired behavior once it is deployed.

In the manufacturing domain, RL has most commonly been applied to problems in process control, scheduling and dispatching, design, quality control of diverse manufacturing systems such as additive manufacturing, semiconductor manufacturing and biological manufacturing (Park *et al.*, 2019; Ueda *et al.*, 2000; Chung et al., 2022; Panzer and Bender, 2022; Li et al., 2023). For process control, applications span chemical processes, welding, machining, and additive manufacturing. For example, Chung *et al.* (2022) proposed a RL method that conducts online learning while incorporating prior knowledge to mitigate defects in fused filament fabrication. In the context of assembly, the focus has largely been on simple insertion tasks (peg-in-a-hole) performed by a robot with some kind of sensory equipment such as vision or force feedback (Panzer and Bender, 2022; Li *et al.*, 2023). Although this represents a notable challenge, it is only one of many problems that arise in assembly processes. On an assembly line, there are many steps that contribute to the overall quality of the final product. One problem that naturally arises is to determine a suitable assembly sequence, a problem that De Giorgio *et al.* (2021) sought to address with RL. Missing from previous work, however, are instances in which parameters may need to be adjusted in response to part variations, e.g., flexible fixtures in automotive, shipbuilding, and aerospace applications. This article explores this issue with the application of assembling composite fuselage sections. As far as we know, this is the first work of using RL for quality control of advanced aircraft assembly.

3. RL for fuselage shape control

In this section, we first formulate the problem of dimensional shape control of aircraft fuselages. From there, we convert the problem into a RL framework, which offers a different way to achieve the goals of quality optimization.

Towards this end, we propose a reward function that captures the optimization objective and sets up an architecture that enforces constraints on the actions that the agent can perform in its environment. Finally, we propose a modified PPO algorithm to solve the RL problem.

3.1. Program formulation of dimensional shape control of aircraft fuselages

In the aircraft assembly process, dimension gaps inevitably exist between two sections of the fuselage due to shape deformation. Therefore, a set of actuators is placed on the boundary of the fuselage to adjust its shape to minimize the dimension gap. The effect of shape adjustment is jointly determined by the layout of actuators along with their forces (push or pull), which is not trivial to optimize. Therefore, we propose to formulate the problem of fuselage shape control as an optimization problem and develop a tailored RL algorithm to solve it. The objective is to generate the layout of actuators over the fuselage along with the applied forces, such that the shapes of two assembled fuselages match well.

Relying on a finite element (FE) model of a fuselage (or possible digital twin in the future), we can set up the shape control problem as an optimization problem. Suppose in our FE model, we define a set $\mathcal{N}$ that contains all nodes of interest to us. These nodes could correspond to a set of key measurement points on a physical part (i.e., the fuselage). Let N denote the number of nodes in $\mathcal{N}$. In our case, by measuring the distance between the opposite nodes in the set $\mathcal{N}$, the finite element model can generate the shape of the fuselage. In addition, there is a total number of n_c candidate locations (denoted as $\mathcal{P}_c = \{P_1, P_2, ..., P_{n_c}\}$) at which the actuators can be applied. However, not all the available positions will be used as (i) the total number of available actuators is much smaller than the available locations, and (ii) the design space of actuator forces will be more complex with the increase of its number. Therefore, we introduce n_a to denote the number of actuators (i.e., the number of candidate locations that will be used). In addition, we use $n_a \ll n_c$ to indicate the fact that the number of available actuators is usually much smaller than the total number of candidate locations. In the setting of the FE model, applying a specific layout of n_a actuators along with designed forces is implemented as applying n_a nonzero forces as boundary conditions out of n_c candidate locations. The forces at the remaining $(n_c - n_a)$ locations can be set to zero, corresponding to no actuator being present there. Hence, we can define a vector $\mathbf{F}$ with the length of n_c where each element F_i corresponds to the force applied at the candidate location $P_i \in \mathcal{P}_c$. The resulting nodal displacements by actuator forces can be evaluated at the N measurement points, which are denoted as $\mathbf{U} = (\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_N})$. These displacements may be nonlinear in response to the applied forces, and they also depend on the fixtures for which boundary conditions have been defined. The nodal displacement at each measurement point is stored as $\mathbf{u_i} = (u_x^i, u_y^i, u_z^i)$ for $i = 1, 2, ..., N$. The objective of shape control is to adjust the dimensions of the fuselage towards the target one. Therefore, there is a corresponding target position of each measurement point, which is denoted as $\mathbf{p_t^i} = (x_t^i, y_t^i, z_t^i)$ (subscript t indicates "target"). Suppose the final positions after force vector $\mathbf{F}$ has been applied are $\mathbf{p_f^i} = (x_f^i, y_f^i, z_f^i)$ for $i = 1, 2, ..., N$, which are obtained by adding the nodal displacements to the initial positions (i.e., $\mathbf{p_f^i} = \mathbf{p_0^i} + \mathbf{u_i}$). The $\mathbf{p_0^i}$ with subscript 0 indicates the initial position at the measurement point i. Note that $\mathbf{U}$ is the output of the FE method with the forces $\mathbf{F}$ as boundary conditions. At each measurement point, the difference between current and target dimensions ϵ can be calculated as follows:

$$\epsilon_{\mathbf{0}}^{\mathbf{i}} = \mathbf{p_0^i} - \mathbf{p_t^i} \tag{1}$$

$$\epsilon_{\mathbf{f}}^{\mathbf{i}} = \mathbf{p_f^i} - \mathbf{p_t^i} = \epsilon_{\mathbf{0}}^{\mathbf{i}} + \mathbf{u_i} \tag{2}$$

From this, we can define initial and final deviation patterns $\mathbf{\Delta_0} = (\epsilon_0^1, \epsilon_0^2, ..., \epsilon_0^N)$ and $\mathbf{\Delta_f} = (\epsilon_f^1, \epsilon_f^2, ..., \epsilon_f^N)$. These deviations characterize the assembly gap between two components that are to be joined, where the shape of one of the components is set as the target shape of the other.

Based on the deviation patterns, we can define an objective function $\mathcal{L}$ to set up a minimization problem. For example, we can choose the Root-Mean-Square Gap (magnitudes of deviations) between the mating fuselage sections (RMSG):

$$e_{RMSG} = \sqrt{\frac{1}{N}(\mathbf{\Delta_f}^{\top}\mathbf{\Delta_f})} \tag{3}$$

In practice, it is easier to avoid taking the square root when solving an optimization problem. We can choose a squared objective function

$$\mathcal{L} = \mathbf{\Delta_f}^{\top}\mathbf{\Delta_f}. \tag{4}$$

When minimizing the RMSG, the actuator forces are typically limited by engineering requirements. Let us denote the lowest and highest allowed limits of forces as F_{LL} and F_{UL}. In addition, there is a maximum of n_a actuators available to use, which is formulated by the second constraint that the total number of non-zero elements in force vector $\mathbf{F}$ can not exceed n_a. We can now formulate a quality optimization problem of composite fuselages as follows:

$$\begin{aligned} &\min \mathcal{L} \\ &F_{LL}^i < F_i < F_{UL}^i \ \forall i = 1, 2, ..., n_c \\ &||\mathbf{F}||_0 \le n_a \end{aligned}$$

Here, the first constraint is straightforward to enforce. However, the second constraint introduces a lot of complexity to the problem because the L_0-norm is NP-hard and difficult to use in solutions. Possible approaches are to reformulate with Boolean variables (i.e., convert to a mixed-integer programming MILP), or to convert and solve with an L1-heuristic. In effect, the latter approach is taken by Du *et al.* (2019), who reformulate the objective with a sparse penalty term and solve the resulting optimization problem via the method of alternating direction method of multipliers.

3.2. RL for dimensional shape control of aircraft fuselages

Model-free RL algorithms have been successfully used to solve many optimization problems (Bertsekas, 2019). A major advantage of RL algorithms over optimization techniques is that they do not rely on solving a system of equations, but rather can interact with an environment (this can be the real world or digital twin or a model), and learn from that interaction. However, it can be challenging to achieve a desired objective and to explicitly enforce constraints on the actions the RL agent can take.

To describe the fundamental steps, a RL problem is set up as follows: there is an environment with a state space $\mathcal{S}$ which is in some state $s \in \mathcal{S}$. From this state, an agent collects an observation o from the environment. This observation may, or may not, be a direct measure of the state of the environment, so it is drawn from a set of possible observations $\mathcal{O}$. Hence, the observation space $\mathcal{O}$ may differ from $\mathcal{S}$. Based on the observation, the agent decides to take an action $a \in \mathcal{A}$, where $\mathcal{A}$ is the set of all possible actions, i.e., the action space. Based on the chosen action, the state of the environment transitions to a new state, s', according to a transition function $P(s'|s, a)$. A reward function $R(s, a, s')$ returns a reward r to the agent. It is typically assumed that the goal is to find an optimal policy for a Markov Decision Process (MDP) that maximizes the reward received over a time horizon H and takes into account a discount factor γ. Namely, the optimal policy π^* for an MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma, H)$ is sought. In the following sections, we will illustrate the details of the proposed approach.

3.2.1. Objective function formulation

In the RL formulation, an optimization objective can be captured via the reward function. However, choosing a reward function is a crucial decision in RL that can determine whether or not the agent is able to learn a successful policy to improve the quality management of aircraft assembly. Furthermore, the reward function can strongly influence the rate at which the policy converges towards a successful policy. A poorly designed reward function can drive an agent towards undesired and unexpected behaviors. For example, if an agent tends to receive a lot of negative rewards during episodes, it may lose its “will to live” and strive towards early episode termination by taking only “bad” actions in quick succession. Another challenging circumstance arises when rewards are rarely given, i.e., rewards are sparse. In this scenario, the agent must explore the environment a lot until it observes a positive reward, and thus, learning tends to be very slow and sample-inefficient. Ideally, rewards are given frequently to provide enough of a learning signal to the agent during training and scaled in a way to not encourage erratic (=too large) policy changes. In situations where a reward function is directly accessible, e.g., scores in Atari games, some successful strategies have been devised that seek to normalize and scale rewards to facilitate learning (van Hasselt *et al.*, 2016; Schaul *et al.*, 2021).

In manufacturing, directly accessible reward functions exist as well. Oftentimes, a deviation from a target value or setpoint is of interest, and the goal is to minimize it. This could be the case for shape errors in mechanical applications (machining, joining, 3d printing) or control of process parameters such as temperature, flow rate, or forces. For this purpose, some distance function is typically used, most often the absolute or squared error, e.g, the L1 or L2 norm.

However, it is not necessarily best to directly use the absolute error term as a reward for an RL agent that is used in the manufacturing system. Suppose we simply return the magnitude of the reduction in error, i.e., $R = e_i - e_f$, where e_i and e_f indicate the dimension errors before and after applying the forces. Since the best achievable error reduction is capped by the initial error (i.e., the ideal case is always to have the initial error reduced to zero), the magnitude of the reward is jointly determined by the performance of the policy and the initial error. If the initial error is small, an ideal policy (reduce the error to zero) can only achieve a small reward. Conversely, if the initial error is very large, even a comparatively bad policy (fail to reduce the error to zero) can still receive a considerable reward, as the error reduction might be large. In an actor-critic setting, where the critic aims to learn the value function (i.e., the cumulative expected future reward on the current state), the issue of using absolute error term as the reward function may slow learning progress. Considering the compatibility of our RL environment with a wide variety of algorithms, we propose the following scheme that normalizes the scale of the rewards to the range [-1,1] in order to facilitate the learning process while maintaining a sensible interpretation of what the reward represents.

For quality targets in a production environment, we propose to calculate rewards based on a *relative improvement* of the distance from the target value relative to the starting distance. Intuitively, we aim to provide zero reward if no change is detected, a positive reward for an improvement, and a negative reward for an increase in error. As for the scale of the rewards, we choose a maximum reward of $+1$ to be returned when the final distance from the target value becomes zero. On the other hand, when the outcome of the episode is worse than, or equal to, a chosen threshold, we bound the negative reward at -1. This was implemented by a simple clipping operation on the reward function. Note that we could also perform a “squishing” transformation, but for a policy that is quite “bad,” such refinement is not really necessary. A straightforward way to formulate this mathematically is as follows:

$$R = \max\left(1 - \frac{e_f}{e_i}, -1\right). \tag{5}$$

In this equation, e_i and e_f refer to the dimension errors calculated via the chosen distance function before and after applying a specific force, respectively. The reward function is visualized in Figure 1. For zero final error, the reward is $+1$. When the initial and final errors are equal, zero reward is given. In the case where the final error is greater than double the initial error, the reward is clipped at -1.

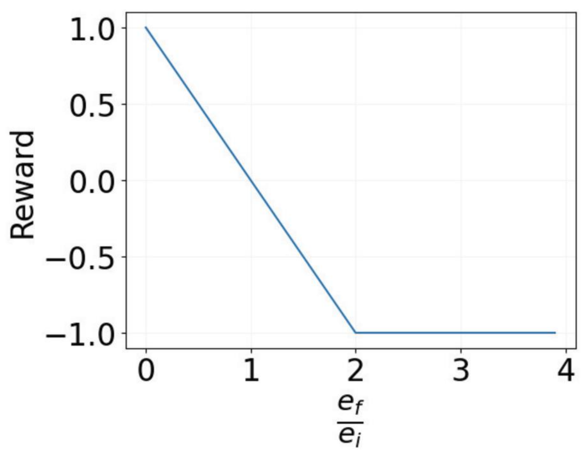

Figure 1. Visualization of the reward function.

Otherwise, the reward is calculated on a linear scale, normalized by the initial error.

3.2.2. Constraints definition

1. Limits on the number of actuators. It is common for an agent to encounter situations where it is impossible to take certain actions. For example, in a game-type environment, rules often dictate what a player is and is not allowed to to. Or in navigation tasks, the presence of a wall may preclude an agent from moving in one or more directions. One may expect the agent to learn such rules or features of the environment by trial and error. However, this would come as a cost since the agent will have to interact with the environment many times to learn when to avoid a set of actions. It also turns out that it is not always easy to design a reward function or other mechanism that provides the necessary signal to the agent from which it can learn the desired behavior. Methods such as providing negative rewards for invalid actions or ignoring invalid actions inside the environment do not always work.

A recent advance to overcome this challenge has been introduced by Vinyals *et al.* (2019), who propose a method to mask out invalid actions when the action space is *discrete*. The method works by setting the probabilities of taking invalid actions to zero, which is done inside the policy neural network. Thus, the agent cannot attempt an invalid action at all. Effectively, this means that the policy utilizes prior knowledge about the environment that is coded directly into the inner workings of the agent. This approach can speed up learning significantly compared with an agent that tries to execute invalid actions over and over. After all, the mask is a function of the state of the environment and, as such, contains information that is very helpful to the agent. Importantly, the neural network weights associated with taking particular invalid actions are not being updated when it is masked out.

Here, we propose an approach for incorporating the influence of constraints when the action space is *continuous*. Such scenarios arise when a continuous action is only allowed to be triggered in a particular situation and is supposed to be at a constant value otherwise (e.g., zero). In general, we can provide a Boolean input mask along with constant replacement values for the continuous output to the agent. As such, the underlying principle of operation is quite similar to that of the discrete case: provide an output mask and replace relevant values at the output of the agent.

A related scenario arises when the allowed number of actions is restricted. Rather than providing a set of *specific* actions to mask out, we may limit the count of continuous actions being output. For example, say a humanoid robot worker has two hands, but suppose there are five hand tools to choose from for assembling a certain part. We can say that the continuous outputs for the tools not in use are to be zero, while the chosen tools require a non-zero output so that they can be activated with the proper parameters. To accomplish this, we can perform a thresholding operation that masks out all but the actions with the largest magnitude, i.e., the actions that the agent most strongly wants to perform. For this to work, the magnitudes of all action outputs are scaled to the same interval of $[-1, 1]$, with the assumption that an appropriate mapping of these output values is performed inside the environment.

2. Limits on the magnitude of forces. We deliberately apply a tanh activation function to the final layer in our neural network. This is in contrast to many implementations of neural network architectures for RL, where the final activation layers are simply fully connected linear layers. The outputs of such layers are unbounded, which can lead to problems when, in contrast, the action space is supposed to be bounded. The naive solution is to simply clip the output of the neural network to match the bounds of the action space. Indeed, this is often done in popular implementations of RL algorithms. However, doing so discards information: suppose the output is constrained to be at most one according to action space bounds. When clipped to this value, the environment reacts the same whether the agent outputs 2, 10 or some other value greater than 1. Hence, it is better to perform a "squishing" operation as can be accomplished with a *tanh* activation, which squeezes the output into the interval $(-1, 1)$ but maintains a gradient that can affect policy updates. From this interval, the force bounds can be enforced by transforming the neural network outputs as follows:

$$F_i = Y_i\left(\frac{F^i_{UL} - F^i_{LL}}{2}\right) + \frac{F^i_{UL} + F^i_{LL}}{2} \quad (6)$$

In this linear transformation, Y_i are the ith outputs from the tanh activation, and F^i_{UL} and F^i_{LL} are the upper and lower limits of permitted forces, respectively. We apply this scaling inside our RL environment.

3.3. Modified PPO

Policy gradient algorithms aim to find a local maximum of a policy objective function $J(\theta)$. The policy parameters θ are changed according to the gradient ascent update rule $\theta_{t+1} = \theta_t + \alpha\nabla_\theta J(\theta)$. In this expression, α is a hyperparameter that corresponds to the step size of policy parameter updates, and $\nabla_\theta J(\theta)$ is the policy gradient. For the most straightforward objective function that underlies the REINFORCE algorithm (Williams, 1992) (also referred to as "vanilla" policy gradient), the objective function chosen is such that the gradient is

$$\nabla J(\theta) = \mathbb{E}_\pi\left[\frac{\nabla\pi(a|s,\theta)}{\pi(a|s,\theta)}\hat{A}\right].$$

This objective directly aims to maximize the future returns under the current policy. It gives rise to the update rule

$$\theta_{t+1} = \theta_t + \alpha G_t\frac{\nabla\pi(a|s,\theta)}{\pi(a|s,\theta)},$$

which intuitively increases the probability of taken actions that increase returns and vice versa. Since REINFORCE was introduced, more sophisticated objective functions have been proposed to improve upon the performance when utilizing policy gradients. As one of the latest advances, PPO (Schulman *et al.*, 2017) improves upon the vanilla policy gradient in several ways. Realizing that policy updates are often too large, PPO simplifies the objective of its direct successor, trust region policy optimization (Schulman *et al.*, 2015) by introducing a clipped objective function:

$$L^{\text{CLIP}} = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\right]. \tag{7}$$

In this expression, $r_t(\theta)$ is the probability ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}.$$

The clipping ratio prevents the probability ratio from moving outside the interval $[1-\epsilon, 1+\epsilon]$, thus limiting how much action probabilities can change for the new updated policy relative to the old one. The min operation makes it so that the clipping is in effect on only one side, depending on the sign of the advantage estimate $\hat{A}$.

To encourage exploration, we can add an entropy loss of the form $L^S = S[\pi_t](s_t)$ scaled by some factor c_s, in which S indicates the entropy function. Thus, the overall objective becomes

$$J(\theta) = L^{\text{CLIP}} + c_s L^S \tag{8}$$

For estimating the advantage function, we use a neural network that acts as a critic. Its parameters are updated to estimate the value function according to the quadratic value loss $L^{VF} = (V_\theta(s_t) - V_t^{\text{target}})^2$. From the value function estimate, the advantage can be estimated for a predetermined period of T timesteps as $\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \ldots + \gamma^{T-t+1}r_{T-1} + \gamma^{T-t}V(s_T)$. If the policy and value networks were to share network parameters, then this objective could be added to equation (7). In our network architecture, they are independent of each other and their objectives can thus remain separated.

3.4. Annealing the variance of the action distribution for efficient exploration

Our proposed RL algorithm aims to explore the available actions (the action space) in order to develop its policy. In continuous action spaces, it is designed to select actions based on a Gaussian policy that is parameterized with a mean μ and a standard deviation σ. Accordingly, on-policy algorithms often employ neural network architectures that estimate these hyperparameters (i.e., μ and σ), in effect having them as the output from a learnable neural network. However, limited trials focused on dynamically adjusting these hyperparameters to guide the exploration of action space and facilitate efficient training. Intuitively, a large value of σ indicates a wide spread-out of the distribution over possible actions, which tends to encourage exploration. On the contrary, a small value of σ indicates the RL agent is more confident about the current policy while is less intent for exploration. Therefore, the propensity of exploration can be tuned by controlling over the standard deviation σ during training. The principle guideline is that the initial training stages should encourage the exploration of action space (a larger value of σ) while the action policy should gradually converge as the training evolves (a smaller value of σ). Hence, we propose to exert greater control over the parameter σ during training. Compared with the common practice of estimating the σ directly from the neural network, our proposed method improves the efficiency and performance of the learning process. In addition, the classic design of the PPO algorithm includes an entropy term in the objective function to control the exploration process. The impact of this entropy term is adjusted through a coefficient c_s (shown in equation (8)), which also requires extensive tuning. Introducing annealing the action distribution variance can eliminate the need to use the entropy term and, therefore, reduces the need for tuning the coefficient. The technical details are provided as follows.

In order for the agent to learn about its environment, in general the agent must explore its action space during training. In the case of the on-policy PPO algorithm, the agent typically is set up to parameterize an action distribution, i.e., a probability distribution from which actions are sampled. In typical implementations of PPO, the agent specifies the mean μ and variance σ^2 for a normal distribution based on the observed state at its input. The actions are then sampled from this distribution. Hence, the agents actually performs the following function: $f: s_t \to \mu, \sigma$ Based on this mapping, the action distribution is

$$\pi(a_t|s_t) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}$$

from which actions are sampled as $a_t \sim \mathcal{N}(\mu, \sigma^2)$.

There are some scenarios where it is desirable for the agent to not be entirely predictable. For example, in a competitive game setting such as tennis, it may be beneficial for the agent to keep the opponent guessing as to where the next shot is going to be aimed. In contrast, engineering environments most often require that the agent acts deterministically, meaning that the same observation produces the same action. Questions about the trustworthiness of a manufacturing process could arise if actions were not repeatable, in particular if the process is subject to validation procedures, as is the case in aerospace applications.

Suppose we require that after training, the RL agent acts deterministically. Then the agent should be tasked to learn the mapping $f : s_t \rightarrow \mu$, from which we can then specify actions as $a_t = f(s_t) = \mu$. This implies that the agent does not have to learn how to specify σ in order to be deployed. Furthermore, it begs the question if learning σ may be detrimental to the final goal, as it may detract from the overarching objective. That is, the agent ought to maximize rewards under a *deterministic* policy, but it acts stochastically during training, and the randomness of actions is, in fact, controlled by σ. It may be the case that a small σ (corresponding to deterministic behavior) maximizes the reward during training, but this is not necessarily guaranteed. So, instead, we propose to gradually guide the actions of the agent from random towards more deterministic behavior by imposing an annealing schedule on σ. For example, we may impose a linear or exponential decay on the variance of the action distribution. Figure 2 illustrates how, as a result, the action distribution changes as training progresses. The agent starts off exploring the environment through stochastic actions. Then, as training progresses, the agent reduces the randomness of its policy according to an annealing schedule for the standard deviation of the action distribution. Hence, during training, the agent is steered towards the deterministic behavior that it is supposed to follow upon being deployed in the real world. In effect, our method reduces the need to tune a hyperparameter that is typically not even accessible for tuning.

There is another reason why we may want to manually set the variance of the action distribution during learning. That is, it may be prudent to ensure cautious exploration so as not to exceed the engineering limits of the assembly. This is especially true if the agent is allowed to learn online rather than inside a simulated environment. Furthermore, a learning strategy that starts with small adjustments centered around zero is a sensible heuristic strategy similar to how an engineer would approach an unknown problem. Taking this a step further, engineering knowledge could be incorporated into a prior for the action distribution.

4. Case study on fuselage shape control in aerospace manufacturing

4.1. Overview

For our case study, we are considering the assembly of a composite fuselage. In our application, two fuselage sections are to be joined together. Due to manufacturing variations, there typically exist gaps between the components that need to be minimized to meet quality targets. Such gaps have been found to be up to an inch in magnitude, far above tolerable limits (Wen *et al.*, 2018). Hence, it is almost always necessary to correct the shape of mating parts. In current practice, shape adjustments are performed via a fixture that features several force-controlled actuators that can exert a force (push or pull) perpendicular to the fuselage skin. This solution is not as straightforward as it seems because it is difficult to prescribe a suitable set of forces. Typically, this requires the input of expert knowledge and several steps of trial-and-error adjustments. Not only is it costly to involve an application engineer for this purpose, but incremental adjustments take a lot of time because they necessitate repeated measurements of the components. Accordingly, it

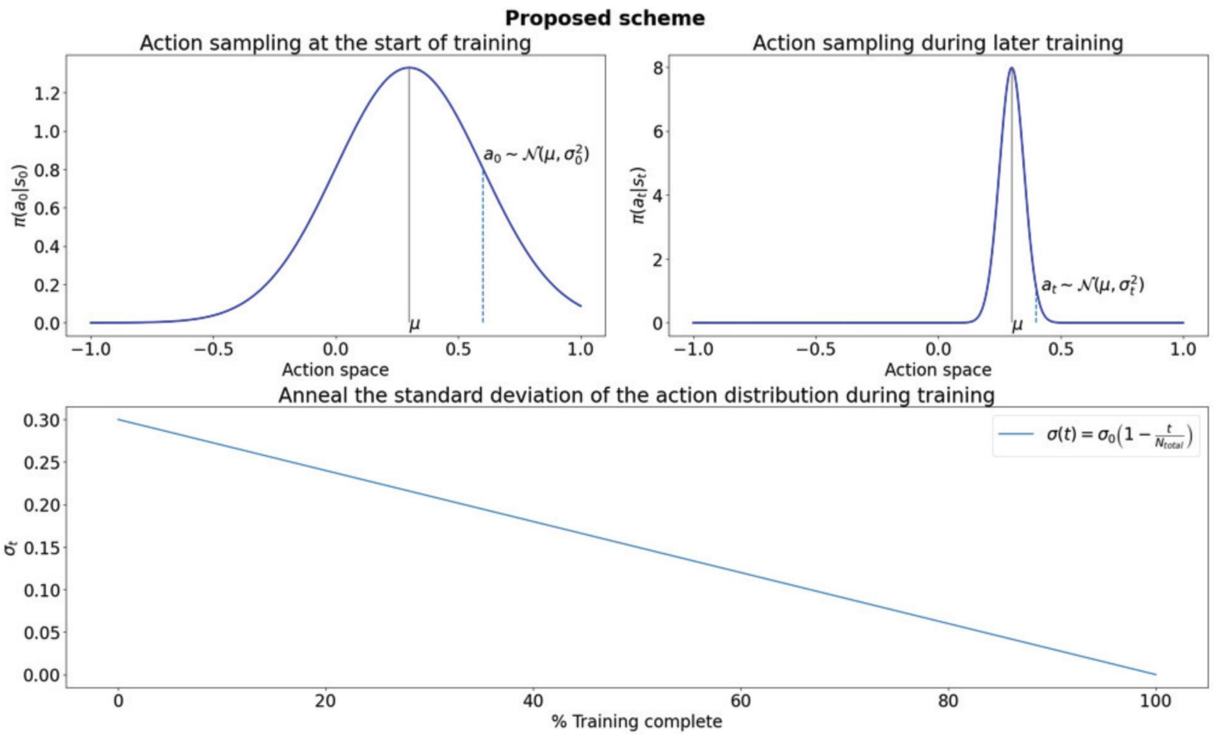

Figure 2. The action distribution changes as training progresses.

is desirable to calculate an optimal set of actuator forces and locations from a single measurement step.

4.2. Problem definition

We define an idealized fuselage shape as a cylinder with a diameter of 216 inches and 288 inches in length. Our model specifications were adapted from Wen *et al.* (2018), who experimentally validated the simulation outputs. Based on a design-of-experiment, we generate a set of deviated fuselage shapes that are generated by applying forces to our nominal ("ideal") starting shape. We use the same design of experiment as Du *et al.* (2019) to generate 40 sample shapes for training and 20 sample shapes for testing purposes. Pairs of these deviated shapes are then to be mated to each other so that the total set of possible scenarios consists of all pairwise combinations of these shapes. In total, there are 1560 training scenarios and 380 test cases.

Mimicking an existing process, we assume that one fixture simply holds one of the two fuselage sections stationary. For the second fuselage section, another fixture is equipped with actuators that can exert forces at up to 18 candidate locations. There are supports and a strap that hold the fuselage in place, as illustrated in Figure 3. According to engineering knowledge, the maximum force that the actuator can apply to the fuselage is 1000 lbs (Wen *et al.*, 2018). Therefore, the action space is defined with the maximum force magnitude as 1000 lbs. In addition, it is desirable to use a small number of actuators when matching the shape of the adjusted fuselage section to that of its counterpart.

4.3. RL environment

The developed RL scheme can be visualized in Figure 4. The detailed simulation platform, surrogate model, action space, observation space, and reward function will be introduced as follows.

4.3.1. Simulation platform

Many RL environments have been created based on to a standardized interface provided by OpenAI gym (Brockman *et al.*, 2016). For this interface, many RL algorithms have been implemented and can thus be used with modifications and setting of hyperparameters. Using the pyMAPDL framework (Kaszynski *et al.*, 2020), we built a customized environment that follows the gym specifications. Our algorithm is adaptable to other environments.

The pyMAPDL package makes it possible to control an existing ANSYS simulation from a Python script. To create our gym environment, we first define a FE model of our fuselage section in ANSYS Workbench using the ACP Pre/Post and Mechanical tools. The fixturing and application of forces are realized via boundary conditions that are applied to the geometrical features of the fuselage. Our simulation encompasses two stages: (i) generation of deviated shapes and (ii) shape adjustment.

We run the first stage of the simulation with input parameters from a design-of-experiment (DOE) to generate a set of solutions that correspond to the initial shapes of mating fuselage sections. From here, the shape adjustment stage is performed. This latter stage is the part of the simulation that the RL agent interacts with - at this point, the agent specifies the actuator forces and locations via boundary conditions defined in the FE model, given the initial observation from the first stage solution output.

4.3.2. Surrogate model

We build a surrogate model according to the approach of previous works (Yue *et al.*, 2018; Du *et al.*, 2019), relying on fuselage deformations generated from a DOE. In total, 40 samples were used for training and 20 for testing purposes. On these samples, the linear model performs very well, as evidenced in Table 1.

Whereas the ANSYS FE model took 7 s to compute a single solution, the surrogate model could provide up to 500+ solutions per second while running on 12 parallel CPU cores. For validation and testing purposes, we rely on the outputs from the ANSYS simulator as our ground truth.

4.3.3. Action space

There are 18 candidate locations for actuators that are used to adjust the shape of the fuselage. The environment can be instantiated with a maximal number of actuators, n_a. The

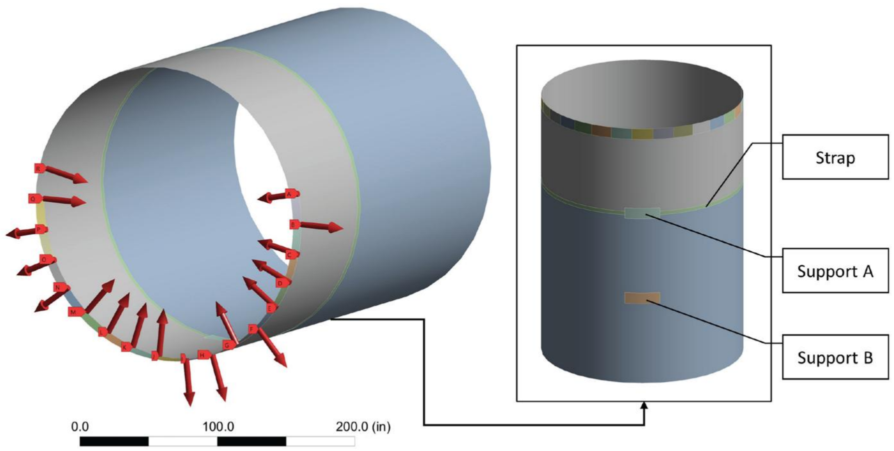

Figure 3. Fixtures and actuator forces in the fuselage model.

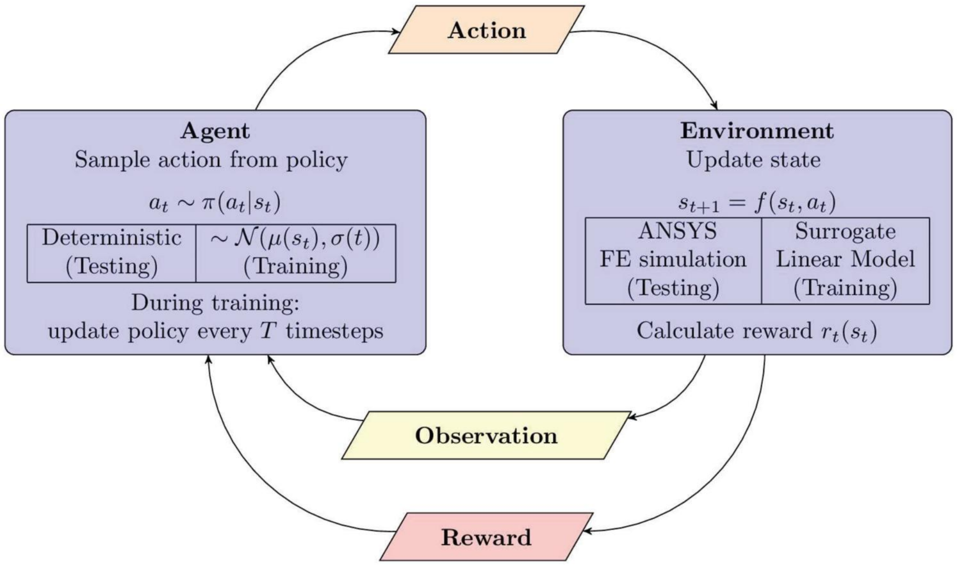
Action
Agent
Sample action from policy
$a_t \sim \pi(a_t|s_t)$
Deterministic (Testing)
$\sim \mathcal{N}(\mu(s_t), \sigma(t))$ (Training)
During training: update policy every T timesteps
Environment
Update state
$s_{t+1} = f(s_t, a_t)$
ANSYS FE simulation (Testing)
Surrogate Linear Model (Training)
Calculate reward $r_t(s_t)$
Observation
Reward

Figure 4. Overview of the RL scheme.

Table 1. Goodness of fit statistics for the surrogate model.

Metric	*Train*	*Test*
MAE [in]	1.18e-8	1.89e-8

agent can specify forces for all $n_c = 18$ actuator positions. However, only the n_a-largest forces are actually applied to the fuselage. In fact, the $(n_c - n_a)$-smallest forces are set to zero by the environment. By doing so, the constraints on the number of actuators is always enforced, regardless of whether or not the agent imposes this constraint on itself.

4.3.4. Observation space

The environment returns the deviations of the fuselage edge from its target shape. In total, deviations at $N = 177$ nodes along the circumference of the fuselage are returned to the agent. The deviation at each node comprises the $x-$ and $y-$ displacements from its target position, measured in inches. The environment returns the observation as a flattened vector with 354 elements.

4.3.5. Reward function

The reward is calculated from the deviations of the final shape as compared with the original shape. More specifically, it is calculated as the relative improvement of RMSG during the shape adjustment according to equations (5) and (3). On the negative side, the reward is clipped at -1 so that the reward function returns values in the interval $[-1, 1]$.

$$R = \max\left(1 - \frac{e^f_{RMSG}}{e^i_{RMSG}}, -1\right) = \max\left(1 - \sqrt{\frac{\Delta_f{}^\top \Delta_f}{\Delta_i{}^\top \Delta_i}}, -1\right). \tag{9}$$

4.3.6. Episode

All forces are applied all at once, hence each episode lasts only a single step.

Table 2. PPO hyperparameters used for training

Hyperparameter	*Symbol*	*Value*
Optimizer learning rate	α	1e-5
Optimizer learning rate annealing		True (linear)
Discount factor	γ	0.99
Max gradient norm		0.5
Minibatch size		16
Number steps		16
Number minibatches		12
Update epochs		4
Number parallel environments		12
Clip coefficient	ϵ	0.2
Entropy coefficient	c_s	various $(0.0, -0.1, -0.01, 0.01)$
Value function coefficient		0.5
Reward normalization		False
Advantage normalization		True

4.3.7. Hyperparameters

The PPO algorithm can be tuned through a number of hyperparameters. For most of them, the default values as recommended by Schulman *et al.* (2017) were used. Some adjustments were made to improve performance in our continuous action space setting. Table 2 provides a summary of the parameters used in training.

4.4. Results and discussion

In this section, the performance of our proposed method is first demonstrated by comparing it with the state-of-the-art benchmark. In addition, the effectiveness of our designed modules is demonstrated through ablation studies.

4.4.1. Actuator selections and experimental results

The desired or allowed number of actuators can be freely set for our learning environment. Intuitively, with more actuators, we expect to have more control over the shape of the fuselage. Figure 5 illustrates the final RMSG after 8,000,000 training steps evaluated over 100 test samples, where the target shape is non-ideal. Indeed, our results show that with a greater number of actuators, a smaller shape error can be achieved. From a cost perspective, we may be interested in the smallest number of actuators necessary to achieve good enough results. As we can see, performance is very similar when eight or more actuators are used but degrades quickly when fewer forces are applied to the fuselage. That is to say, beyond eight actuators, the return of adding more actuators diminishes.

The agent tends to select actuators on the sides rather than the bottom of the fuselages. This can be explained by the fact that, given the fixture supports, the fuselage can move more freely in the horizontal direction than in the vertical direction. Thus, a greater displacement can be achieved with these actuators. Examples of actuator selections and corresponding forces are illustrated in Figure 6, which shows the cross-sectional view of the fuselage joining edge before and after shape control has been applied.

The magnitudes of forces also tend to be greater at locations where larger deviations are present. This is not a surprise. In effect, the results using a small number of actuators show that a few actuators with large forces are sufficient to roughly match the target shape. Additional actuators then help eliminate local gaps via small forces applied in the vicinity where gaps still exist. In a heuristic step-wise strategy, an operator would typically adjust the overall shape with a few actuators and then try to match the overall shape by fine-tuning forces at additional actuator locations where gaps are still present. The downside of the manual step-wise method is that each subsequent addition of a force can distort the overall shape, and so the fact that forces are prescribed in a single step as in our method is desirable.

We compare our performance to forces calculated via the method by Du *et al.* (2019). In this related work, the non-nominal fuselage shapes were adjusted back to the ideal shape. We replicated this approach and evaluated the results on the corresponding test samples from the DOE. As shown in Table 3, the mean RMSG we obtain with our RL agent is smaller than that of the benchmark method. Furthermore, the RL agent achieves a smaller variance of the RMSG

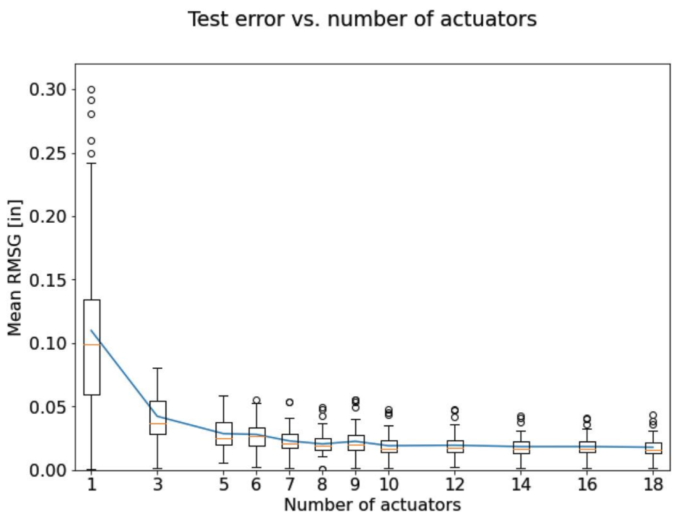

Figure 5. The final test RMSG v.s. the number of actuators.

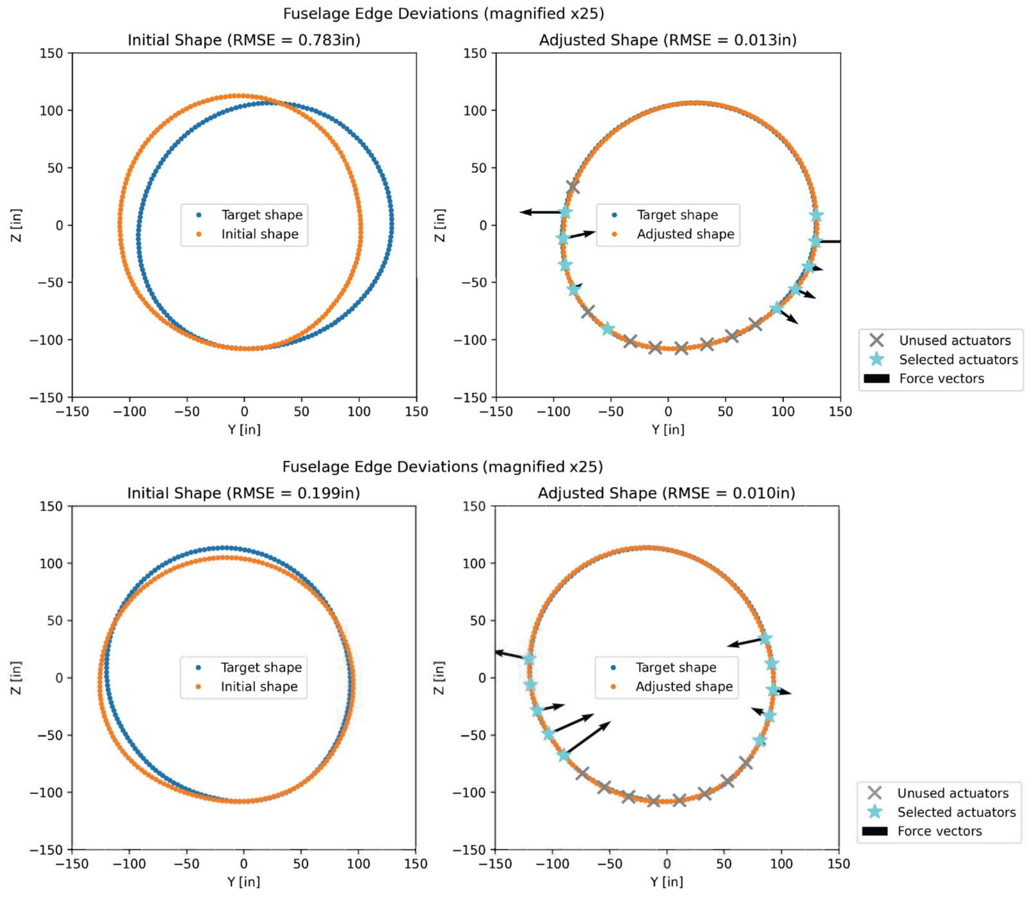

Figure 6. Examples of shape adjustment results, indicating which actuators were chosen. Force arrows are scaled by the magnitude of the proposed forces.

Table 3. Comparison statistics on 20 test samples with $n = 10$ forces applied.

Method	*Mean RMSG [in]*	*Stdev RMSG [in]*	*Mean MF [lbf]*
Du *et al.* (2019)	0.011	0.0063	289.5
RL (our method)	0.010	0.0025	214.8

across the results. At the same time, our method applies smaller maximum forces on the parts, suggesting that the assembly experiences less stress during the assembly process. Hence, our method can achieve overall better performance than the benchmark method. An added benefit of our method is that the trained RL agent performs a single forward pass in the policy neural network given the initial deformation of the fuselage, which is a very fast and simple computation that does not require any interaction with the environment. Contrast this with the benchmark method, which has to re-run thousands of calculations on the surrogate model in order to return a set of actuator forces. Although this may take only a handful of seconds with a surrogate model, it is time-prohibitive to utilize a FE method for the optimization. In turn, the RL agent is actually able to refine its solution with just a handful of interactions with the environment, so it can be feasible to directly act upon the FE model.

4.4.2. Ablation study on the reward function

To demonstrate the effectiveness of our proposed reward function, we compare the performance of our proposed method using the reward function evaluating *relative improvement* (our proposed) versus the existing reward function evaluating the absolute error reduction in different measurement units (i.e., inches and millimeters), which is defined as the difference of dimension errors before and after applying the actuators ($R = e_i - e_f$).

The results are demonstrated in Figure 7, from which we can summarize the advantages of our proposed reward function into two aspects. First, following the discussion in Section 3.2.1, the learning process using the proposed reward function consistently encourages the policy that reduces the initial dimension error towards zero, which is demonstrated to converge faster (i.e., shown in the blue curve in Figure 7) than using the absolute error reduction as the reward function (i.e., shown in the orange and green dashed curves in Figure 7). Second, the proposed reward is invariant to the magnitude of the error and consistently falls into $[-1, 1]$, which enables its invariant performance under different units of dimension measurement and different magnitudes of initial errors. We also demonstrated how the magnitude of the reward will impact the training process. when using the absolute reduction in dimension error as the reward, we compared the results using inches or millimeters as the measurement unit for dimension error. We can observe a smaller magnitude of the reward function (measured in inches and shown in the orange dashed curve in Figure 7) leading to a faster convergence compared with a larger magnitude (measured in inches and shown in the orange dashed curve in Figure 7). It provides another reason to define the reward function as the relative improvement

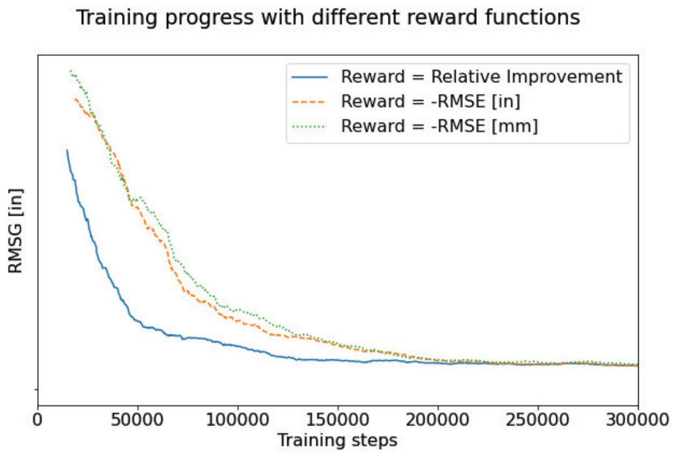

Figure 7. Training process using different reward functions.

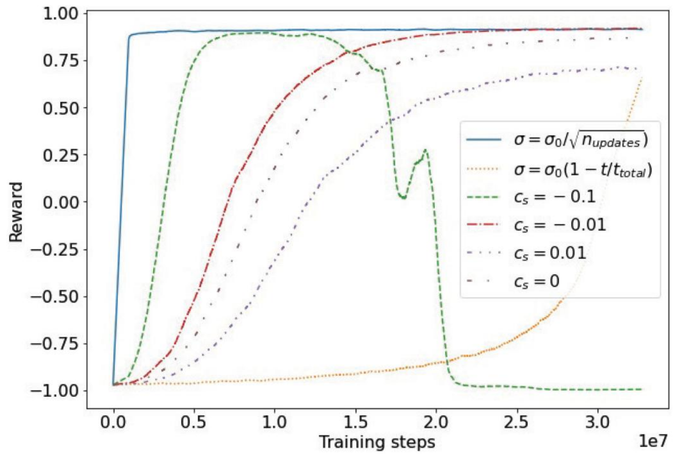

Figure 8. Training progress (rolling average over 1000 update steps) in terms of the episodic return. Controlling exploration via the entropy loss does not achieve the same speed of learning as a well-chosen annealing schedule. If the entropy loss is set to aggressively discourage exploration, a breakdown can occur, e.g., $c_s = -0.1$ in our example.

for a consistently small magnitude for the reward function, which always falls into $[-1, 1]$.

4.4.3. Ablation study on annealing the action distribution variance

To demonstrate the effectiveness of our proposed improved PPO algorithm in our case study, we compare the performance of the PPO algorithm with annealing the standard deviation of the action distribution versus using the entropy term in the objective function (the second term in equation (8)) when controlling the exploration. We find that the variance of the action distribution has a significant effect on the training progress, as evident in Figure 8.

Using the default settings of an initial standard deviation $\sigma_0 = 1$ for initializing the action distribution, training the PPO algorithm without the entropy loss term does not yield satisfactory results for our problem. With this approach, the agent is set to explore far too much, and it takes a long time to sufficiently refine the policy. Furthermore, when we add the suggested entropy loss term to the reward optimization objective, the training progress becomes even worse. The coefficient c_s in equation (8) is the weight associated with

Table 4. Ablation study results comparing the performance of the trained agents on 100 test samples. Our proposed method outperforms the standard implementation of PPO wherein no annealing schedule for the action distribution variance is set.

	Action variance annealing ($\sigma_0 = 1$)		*Standard PPO* (σ *indirectly controlled via* c_s)			
Method	$\sigma = \sigma_0/\sqrt{n_{updates}}$	$\sigma = \sigma_0(1 - t/t_{total})$	$c_s = -0.1$	$c_s = -0.01$	$c_s = 0.0$	$c_s = 0.01$
RMSG [in]	**0.01664**	0.1202	3.167	0.03562	0.03897	0.06147

the entropy term. A coefficient of 0.01 is often prescribed as a good starting point, but in our scenario, this only makes things worse. This is because by adding the entropy term, exploration is further encouraged rather than discouraged, even though, in our case, the opposite turns out to be needed. We can instead make c_s negative, which is an unusual choice. At a small scale ($c_s = -0.01$), we see a slight improvement in how fast training progresses, but pushing this parameter further into the negative, training breaks down. At $c_s = -0.1$, the episodic returns all of a sudden decrease after 14,000,000 time steps and then fall off dramatically. All training progress up to that point is lost. An explanation for this is that the entropy loss term starts to dominate the reward optimization objective function, and the reward itself no longer drives the behavior of the agent. This motivates us to design a new method to control the exploration process more effectively.

Annealing the action distribution variance can intuitively encourage the exploration at the beginning when the variance the large and gradually converges to the best policy with the reduction of variance. Simple linear annealing of the standard deviation of the action distribution according to $\sigma = \sigma_0(1 - t/t_{total})$ is initially very slow to learn a policy that improves the RMSG at all. It takes almost 30,000 time steps for the agent to achieve an improvement over the initial configuration of the pairs of fuselage sections. At this point, σ is around 6% of σ_0. This indicates that the standard deviation of the action distribution is far too large to begin with and points us towards prescribing a more rapidly decaying annealing function for σ. In practice, it may still be worthwhile to initially perform a linear annealing schedule as it effectively sweeps the available range of the action distribution variance. By paying attention to the episodic rewards, a more feasible value for σ_0 may be determined. For example, we may consider $\sigma_0 = 1 - 30{,}000{,}000/32{,}000{,}000 = 0.0625$ as a potential choice in our example.

Once we choose a more sensible annealing schedule, i.e., penalizing the initial action distribution by the square root of the number of policy updates already performed ($\sigma = \sigma_0/\sqrt{n_{updates}}$), the agent exhibits much better and faster learning performance. The policy begins to converge towards $R = 1$ quickly. Table 4 shows that this annealing schedule achieves a better performance on 100 test samples (which are identical across evaluations) than any other approaches.

5. Conclusion

The exploration–exploitation tradeoff during training remains an important consideration for reinforcement learning algorithms. In this article, we proposed a strategy that exerts more direct control over the standard deviation of the action distribution of a PPO agent. Typically, the PPO agent's eagerness to explore is tuned by adding an entropy loss term to the reward optimization objective. However, we showed that this can lead to surprising and undesirable behavior. Our scheme can ensure that the agent explores enough at the beginning of training and then refines its policy while moving closer to acting predictably as training progresses. When deterministic behavior is required upon deployment of the agent in the real world (as is often the case in manufacturing applications), this strategy ensures that the agent was trained in a manner that reflects the desired behavior. By choosing a sensible annealing schedule, it is possible to converge to a well-performing policy in a more sample-efficient manner.

We successfully proposed and implemented an RL algorithm for fuselage shape control in an aircraft assembly case study. The results demonstrate that the developed RL algorithm can achieve high scalability to a large design space, improved adaptability to unseen scenarios, and superior shape control accuracy. Our method is superior when compared with the current practice from three perspectives: (i) it can deal with a large design space to optimize the layout and forces of actuators to adjust the fuselage shape toward a non-ideal target shape, whereas the benchmark method adjusts the fuselage components to their ideal shape; (ii) our method can be directly applied to unseen scenarios by conducting a single forward pass in the policy neural network, whereas the benchmark method has to perform many calculations on its surrogate model or simulation in order to solve the optimization from scratch; and (iii) the case study shows the RL method achieves the best performance on shape control by receiving the smallest shape deviation. Therefore, it can be considered the best approach to revolutionizing advanced aircraft assembly and enhancing the productivity, quality, and safety of aircraft products.

With respect to the limitations of the proposed method, it can be roughly summarized into two folds: (i) restricted by the nature of RL, the proposed method still requires a certain number of samples for training and optimization purposes, which restricts its application on the simulation environment instead of the real manufacturing process considering the potential "trial-and-error" costs; (ii) there inevitably exists gap between the simulation and real manufacturing scenario, which challenges the practical effect of the solution generated from the simulation. However, we would like to highlight that the simulation model used in our work has been carefully calibrated and shown to have an accurate representation of the real manufacturing scenario (Wen *et al.*, 2018).

The proposed method can be regarded as a trial for developing an "AlphaGo" or "AlphaZero" for advanced aircraft assembly, in particular, the fuselage shape control process. The proposed method can also be extended to other manufacturing systems, given that the simulation environments are well-defined. In addition, by targeting the sample efficiency, we can further develop novel RL methods that can better incorporate the domain knowledge into the exploration of action space to learn optimal policy more efficiently and reduce the need for training samples.

Acknowledgments

The authors thank Dr. Zhenyu (James) Kong for the insightful discussions and constructive suggestions.

Funding

The authors acknowledge the funding support from the U.S. National Science Foundation under Award CMMI-2035038 and the Grainger Frontiers of Engineering Grant Award from the U.S. National Academy of Engineering. Dr. Xiaowei Yue was partially supported by the Beijing Natural Science Foundation under ID 3244032.

Notes on contributors

Dr. Yinan Wang serves as the tenure-track assistant professor in the Department of Industrial and Systems Engineering at Rensselaer Polytechnic Institute. He received the PhD degree in the Grado Department of Industrial and Systems Engineering at Virginia Tech. His research interest lies in developing systematic engineering-driven machine learning methodologies with applications to advanced manufacturing. He is a recipient of FTC Early Career Award, 7 best paper awards (e.g., IISE Transaction Best Paper Award), and two best dissertation awards. He also won the Faculty Achievement Award from RPI.

Dr. Oliver Tim Lutz is a Clinical Assistant Professor in the Darla Moore School of Business at the University of South Carolina. He received the PhD degree in the Grado Department of Industrial and Systems Engineering at Virginia Tech. His research interest lies in computational simulation and data analytics for aerospace manufacturing.

Dr. Xiaowei Yue Currently, he is an Associate Professor with the Department of Industrial Engineering, Tsinghua University. Prior to that, he was an Assistant Professor with the Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, USA. His research interests include machine learning for advanced manufacturing. He is a recipient of the IISE Hamed K. Eldin Outstanding Early Career IE in Academia Award, the SME Outstanding Young Manufacturing Engineer Award, the IISE Manufacturing and Design Young Investigator Award. He received the Grainger Frontiers of Engineering Grant Award from the U.S. National Academy of Engineering (NAE). He serves as an Associate Editor for IISE Transactions, and IEEE TASE, and IEEE TNNLS. He is selected to be an Editorial Board Member for PNAS Nexus, an open-access journal of the U.S. National Academy of Sciences (NAS). He is a senior member of IISE, and is selected to be the president-elect of IISE QCRE division.

Dr. Jaime Camelio is currently the Associate Dean for Research, Innovation, and Entrepreneurship in the College of Engineering at the University of Georgia. From 2008 to 2019, he was a professor for Advanced Manufacturing in the Grado Department of Industrial and Systems Engineering at Virginia Tech. He served as Chief Technology Officer at the Commonwealth Center for Advanced Manufacturing (CCAM) from 2016-2019. Dr. Camelio obtained his BS and MS in Mechanical Engineering from the Catholic University of Chile in 1994 and 1995, respectively. In 2002, he received his PhD from the University of Michigan. His professional experience includes working as a consultant in the Automotive/Operations Practice at A.T. Kearney Inc. He has extensive experience in operation management in manufacturing environments including industry 4.0 transformation, production control, and lean manufacturing. His research interests are in innovation education, intelligent manufacturing, and cyber-physical security. He has authored or co-authored more than 120 technical papers. He teaches undergraduate and graduate courses related to design, manufacturing processes, manufacturing systems, and data mining.

ORCID

Yinan Wang http://orcid.org/0000-0002-4079-1658
Xiaowei Yue http://orcid.org/0000-0001-6019-0940

Data and code availability

The data and code of this work are available via the link: https://github.com/otlutz/Fuselage-Actuator-RL.

References

AlBahar, A., Kim, I., Wang, X. and Yue, X. (2022). Physics-constrained Bayesian optimization for optimal actuators placement in composite structures assembly. *IEEE Transactions on Automation Science and Engineering*, **20**(4), 2772–2783.

Bertsekas, D. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific, Nashua, NH.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Cai, H., Zhu, J. and Zhang, W. (2021). Quality deviation control for aircraft using digital twin. *Journal of Computing and Information Science in Engineering*, **21**(3), 031008.

Chung, J., Shen, B., Law, A.C.C. and Kong, Z.J. (2022). Reinforcement learning-based defect mitigation for quality assurance of additive manufacturing. *Journal of Manufacturing Systems*, **65**, 822–835.

De Giorgio, A., Maffei, A., Onori, M., and Wang, L. (2021). Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing. *Journal of Manufacturing Systems*, **60**, 22–34.

Du, J., Cao, S., Hunt, J.H., Huo, X. and Shi, J. (2022). A new sparse-learning model for maximum gap reduction of composite fuselage assembly. *Technometrics*, **64**(3), 409–418.

Du, J., Yue, X., Hunt, J.H. and Shi, J. (2019). Optimal placement of actuators via sparse learning for composite fuselage shape control. *Journal of Manufacturing Science and Engineering*, **141**(10).

Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S. and Hester, T. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, **110**(9), 2419–2468.

Guo, F., Xiao, Q., Xiao, S. and Wang, Z. (2023). Analysis on quantifiable and controllable assembly technology for aeronautical thin-walled structures. *Robotics and Computer-Integrated Manufacturing*, **80**, 102473.

Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P. and Wang, Z. (2023). Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, **35**(7), 8762–8782.

Hollenstein, J., Auddy, S., Saveriano, M., Renaudo, E. and Piater, J. (2022). Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *arXiv preprint arXiv:2206.03787*.

Kaszynski, A. *et al.* (2020). Pyansys: Python interface to mapdl and associated binary and ascii files. *Zenodo. doi 10*.

Lee, C., Wu, J., Wang, W. and Yue, X. (2020). Neural network Gaussian process considering input uncertainty for composite structure assembly. *IEEE/ASME Transactions on Mechatronics 27*(3), 1267–1277.

Li, C., Zheng, P., Yin, Y., Wang, B. and Wang, L. (2023). Deep reinforcement learning in smart manufacturing: A review and prospects. *CIRP Journal of Manufacturing Science and Technology*, **40**, 75–101.

Machado, M.C., Bellemare, M.G., Talvitie, E., Veness, J., Hausknecht, M. and Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, **61** 523–562.

Manohar, K., Hogan, T., Buttrick, J., Banerjee, A.G., Kutz, J.N. and Brunton, S.L. (2018). Predicting shim gaps in aircraft assembly with machine learning and sparse sensing. *Journal of Manufacturing Systems*, **48**, 87–95.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J.W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A. *et al.* (2021). A graph placement methodology for fast chip design. *Nature*, **594** (7862), 207–212.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. *et al.* (2015). Human-level control through deep reinforcement learning. *Nature*, **518** (7540), 529–533.

Mou, S., Biehler, M., Yue, X., Hunt, J. H. and Shi, J. (2023). Spac: Sparse sensor placement-based adaptive control for high precision fuselage assembly. *IISE Transactions*, **55** (11), 1133–1143.

Oliff, H., Liu, Y., Kumar, M., Williams, M. and Ryan, M. (2020). Reinforcement learning for facilitating human-robot-interaction in manufacturing. *Journal of Manufacturing Systems*, **56**, 326–340.

Panzer, M. and Bender, B. (2022). Deep reinforcement learning in production systems: A systematic literature review. *International Journal of Production Research*, **60** (13), 4316–4341.

Park, I.-B., Huh, J., Kim, J. and Park, J. (2019). A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. *IEEE Transactions on Automation Science and Engineering*, **17** (3), 1420–1431.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R.Y., Chen, X., Asfour, T., Abbeel, P. and Andrychowicz, M. (2017). Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905.*

Raffin, A., Kober, J. and Stulp, F. (2022). Smooth exploration for robotic reinforcement learning, in *Proceedings of Conference on Robot Learning*, London, UK, pp. 1634–1644.

Schaper, D. (2022, Feb) FAA toughens oversight of Boeing's 787 dreamliner. Available at URL:https://www.npr.org/2022/02/15/1080930976/faa-toughens-oversight-of-boeings-787-dreamliner (accessed: 12/18/2022).

Schaul, T., Ostrovski, G., Kemaev, I. and Borsa, D. (2021). Return-based scaling: Yet another normalisation trick for deep rl. *arXiv preprint arXiv:2105.05347.*

Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P. (2015). Trust region policy optimization, in *Proceedings of International Conference on Machine Learning*, Lille, France, pp. 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347.*

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. *et al.* (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, **362** (6419), 1140–1144.

Tokic, M. (2010). Adaptive ε-greedy exploration in reinforcement learning based on value differences, in *Annual Conference on Artificial Intelligence*, Springer, Berlin, Heidelberg, pp. 203–210.

Ueda, K., Hatono, I., Fujii, N. and Vaario, J. (2000). Reinforcement learning approaches to biological manufacturing systems. *CIRP Annals*, **49** (1), 343–346.

van Hasselt, H.P., Guez, A., Hessel, M., Mnih, V. and Silver, D. (2016). Learning values across many orders of magnitude, in *30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P. *et al.* (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, **575**,(7782), 350–354.

Wang, W., Yue, X., Haaland, B. and Jeff Wu, C. (2022). Gaussian processes with input location error and applications to the composite parts assembly process. *SIAM/ASA Journal on Uncertainty Quantification*, **10** (2), 619–650.

Wang, Y., Lutz, T., Yue, X. and Du, J. (2024). Smartfixture: Physics-guided reinforcement learning for automatic fixture layout design in manufacturing systems. *IISE Transactions*, 1–20.

Wang, Y. and Yue, X. (2024). Multimodal deep learning for manufacturing systems: Recent progress and future trends, in N. Gaw, P. M. Pardalos, M. R., Gahrooei (eds.), *Multimodal and Tensor Data Analytics for Industrial Systems Improvement*, Springer, Cham, Switzerland, pp. 221–252.

Wen, Y., Yue, X., Hunt, J. H. and Shi, J. (2018). Feasibility analysis of composite fuselage shape control via finite element analysis. *Journal of Manufacturing Systems*, **46**, 272–281.

Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, **8** (3), 229–256.

Yue, X., Wen, Y., Hunt, J.H. and Shi, J. (2018). Surrogate model-based control considering uncertainties for composite fuselage assembly. *Journal of Manufacturing Science and Engineering*, **140** (4).

Zhang, W., An, L., Chen, Y., Xiong, Y. and Liao, Y. (2021). Optimisation for clamping force of aircraft composite structure assembly considering form defects and part deformations. *Advances in Mechanical Engineering*, **13**(4), 1687814021995703.

Zhong, Z., Mou, S., Hunt, J.H. and Shi, J. (2022). Finite element analysis model-based cautious automatic optimal shape control for fuselage assembly. *Journal of Manufacturing Science and Engineering*, **144** (8), 081009.