CAD Dimensional Reduction for Shell Modeling using Reinforcement Learning

Steven J. Owen* Armida J. Carbajal Matthew G. Peterson Corey D. Ernst

Abstract

This study presents advancements in shell model preparation, enhancing the design to simulation workflow. An approach is introduced to efficiently reduce thin CAD volumes to sheet body equivalents while maintaining geometric and topological integrity. Machine learning techniques, including supervised and reinforcement learning, guide reduction actions. Thin volume assembly reduction for shell modeling employs a multi-agent reinforcement learning algorithm to establish interconnected sheet bodies with designated attributes. Central to this approach is a novel supervised learning method predicting suitable dimensional reduction operations within a CAD tool. Developed tools enable efficient management of thin volume assemblies, contributing to accurate shell model generation with potential applications in design to simulation workflows.

1 Introduction

In this research, we address the task of converting an assembly predominantly composed of thin 3D volumes into a collection of sheet bodies, suitable for meshing with quadrilateral or triangular elements. The exemplar problem central to our study pertains to key transportation systems. The design solid model in this context is constituted by a series of interconnected thin 3D solid volumes.

For efficient modeling of mechanical responses, analysts typically employ shell finite elements instead of complete 3D hexahedral or tetrahedral elements. The challenge here is to convert the 3D set of thin volumes into a network of connected surface manifolds, to which a mesh of triangles or quadrilaterals can be applied.

The process primarily uses commercial CAD software, applying **copy** and **midsurface** operations to convert volumes into a collection of extended and interconnected surfaces, forming a complete shell model. However, crafting such complex models, exemplified in Figures 1 and 2, can consume a disproportionate amount of time relative to the entire simulation workflow, delaying subsequent analysis.

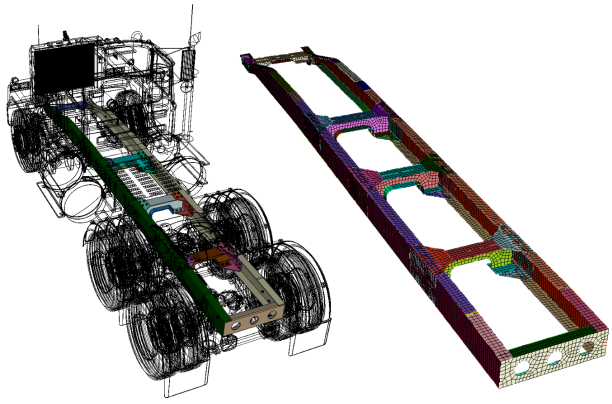


Figure 1: Left: CAD assembly of Mac Superliner illustrating primary structural frame. Right: Shell model of structural frame meshed with quad elements.

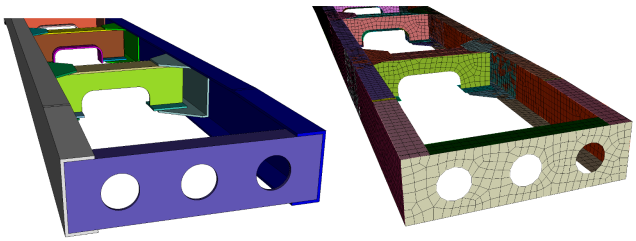


Figure 2: Left: Close-up of 3D model of structural frame in figure 1. Right: Close-up of shell model.

Figures 1 and 2 depict a side-by-side comparison between the original design solid model (left) and the final shell model, primed for analysis (right). In this study, we put forth a novel multi-agent reinforcement learning strategy aimed at developing a comprehensive solution for thin volume reduction, significantly cutting down the time-to-simulation.

The paper begins with a discussion on shell elements, historical approaches, and existing methods for thin volume reduction. A heuristic approach employing geometric reasoning serves as a control, against

*Sandia National Laboratories, Albuquerque, NM. sjowen@sandia.gov, ajcarba@sandia.gov, mgpeter@sandia.gov, cdernst@sandia.gov. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

which we evaluate subsequent reinforcement learning (RL) methods. A novel supervised learning model for thin volume reduction prediction is introduced, alongside a graphical tool for analyst support. The paper then presents a multi-agent RL procedure utilizing supervised learning to address assembly reduction, illustrated with real-world examples.

2 Background

2.1 Shell Finite Element Analysis Finite element analysis utilizing beam and shell elements is a fundamental technique for modeling linear elastic responses when dealing with domains containing one or two "thin" dimensions [1][2]. Zienkiewicz [3] underscores the rationale for using beams and shells:

> Structures with one dimension significantly smaller than the other two define plate and shell problems. A *plate* exhibits a single thin dimension, known as its thickness, while a *shell* is curved in space and features such a small thickness direction. Structures with two small dimensions are labeled as *beams*, *frames*, or *rods*. Accurately solving linear-elastic problems with one (or more) small dimension(s) using standard three-dimensional finite element formulations is generally inefficient due to numerical ill-conditioning, impeding accurate solutions of the resulting algebraic equations.

Beam and shell finite elements remain crucial for modeling linear elastic responses in structural analysis, particularly for systems like transportation, which encompass elements characterized as plates or beams with thin dimensions. This study focuses solely on the shell problem with one thin dimension, recognizing that representing both beams and shells within the same model presents a limitation. Despite this limitation, our research intentionally concentrates on shells, while beams are reserved for future exploration.

In the context of thin volume assemblies, typically represented as intricate 3D models, the process of *dimensional reduction* is essential. This process involves converting a 3D parent volume into one or more manifold surfaces or curves. To achieve this, various methods, such as the *medial axis* [5][4], have been developed. The medial axis method generates a skeleton representation by tracing equidistant points from surfaces [6]. Suresh [7] further advanced this approach, developing a skeletal dimensional reduction procedure for extracting shell models for finite element analysis. Additionally, these methods have been applied to address decomposition problems for hex meshing, where the skeleton aids in geometric reasoning and geometric decomposition [8].

Despite their theoretical appeal, medial axis methods have shown limited robustness in complex cases, often introducing non-manifold surfaces or "wings." This necessitates additional steps for effective shell modeling. Moreover, fully automatic approaches often lack the incorporation of engineering domain knowledge, which is crucial for meeting precise component requirements. Analyst tools typically offer both automated and graphical adjustments to optimize manifold surface creation.

Ansys SpaceClaim [9] is among a handful of, popular commercial tool, offering automated mid-surface generation and integration with beam and shell construction. Notably, the proprietary methods used for constructing shells in Ansys SpaceClaim have not been publicly disclosed. While it excels in preparing Ansys solid mechanics analysis input decks, analysts often resort to manual methods for inspecting, reducing, and extending volumes, a time-consuming process prone to errors and iterative adjustments to meet constraints and achieve optimal results.

3 Supervised Learning: Predicting *Reduce* Solutions

There has been considerable progress recently in geometric processing utilizing AI methodologies, as seen in [10]. However, to our knowledge, no existing work specifically targets the dimensional reduction problem for thin volumes. Our method leverages a Reinforcement Learning (RL) strategy that builds on the foundation of our previous Supervised Learning (SL) model, detailed in [11] and [12]. This SL model is adept at characterizing geometric CAD volumes, a critical aspect of our RL framework. Previously, we successfully employed ensembles of decision trees (EDT) [18] for accurate CAD part classification, a technique that greatly influences our current methodology. Our current research marks an innovative stride by integrating RL for dimensional reduction in CAD models, introducing a unique perspective to the field.

The Cubit® Geometry and Meshing Toolkit [20], developed at Sandia National Laboratories, offers a robust foundation for tool development. It provides an extensive set of CAD operations accessible through a straightforward command syntax and a comprehensive Python API. Additionally, it leverages the capabilities of the third-party library, ACIS [21], to facilitate geometry modification procedures

We establish two fundamental models focusing on key Cubit® CAD operations: **copy** and **midsurface**. In the **copy** operation, a thin volume is represented by parallel surfaces separated by a uniform, small thickness. This method selects a continuous set of surfaces to serve as the reduced sheet body representation of

the 3D volume. In contrast, the **midsurface** operation identifies the central manifold surface between two opposing sets of surfaces. It's worth noting that a generalized midsurface operation goes beyond the scope of this work due to the inherent complexities involved in creating general midsurfaces for arbitrary geometries. Instead, we focus our midsurface application on simple thin-walled structures where the identification of the midsurface is straightforward.

These primary actions, encompassing two **copy** operations and one **midsurface** operation, are visually depicted in Figure 3, showcasing how they transform 3D thin volumes into sheet bodies. Figure 3 also illustrates the connections between neighboring volumes, emphasizing that these neighborhood interactions influence the selection among the three solutions. Furthermore, in more complex cases like the one depicted in Figure 4, multiple surfaces can contribute to the **copy** operation.

Supervised machine learning uses a training dataset $(\mathbf{x_1}, \mathbf{y_1}), ..., (\mathbf{x_n}, \mathbf{y_n})$, where $\mathbf{x}$ represents input features, $\mathbf{y}$ represents labels, and f in $\mathbf{y} = f(\mathbf{x})$ signifies the mapping from input features to labels.

Features ($\mathbf{x}$) consist of fixed-length vectors of scalars for both **copy** and **midsurface** operations, categorized into three groups. The initial category covers volume characteristics like shape, size, and geometry, as detailed in previous research [12] . Subsequent categories focus on the chosen operation (**copy** or **midsurface**), capturing local surface nuances. The third category includes contextual information, such as shared surfaces and connectivity with neighboring volumes.

Labels ($\mathbf{y}$) for both **copy** and **midsurface** models represent a *suitability* score. A score of 1 indicates an ideal operation fit, while 0 represents a poor fit. Typically, the suitability score is determined automatically by the RL procedure's reward value, but manual assignment is possible.

Our SL model, similar to previous work, employs an ensemble of decision trees (EDT) using scikit-learn [19], a Python-based ML tool. The SL model's role is to select the most suitable reduction operation given a local volume and its surroundings.

We initiate the training data with a small set of CAD models, and the RL algorithm plays a crucial role in collecting ground truth data by applying it to numerous models. Initial training data can also come from user preferences, capturing desired outcomes for thin volume reduction in isolated cases.

4 Reinforcement Learning

4.1 Background Reinforcement learning (RL) is a crucial component of machine learning, commonly applied in areas like *game theory* and *control theory* [13]. Unlike supervised learning, RL doesn't rely on labeled data pairs, but rather on structured rules and rewards in an environment. This empowers an *agent* (primary decision maker) to explore and learn from the environment.

At its core, RL involves:

- State-Space, **S**, where the agent operates.
- Action set, **A**, for the agent's choices.
- Decision Policy, **P**, governing action probabilities.
- Reward Policy, **R**, providing rewards for state transitions.

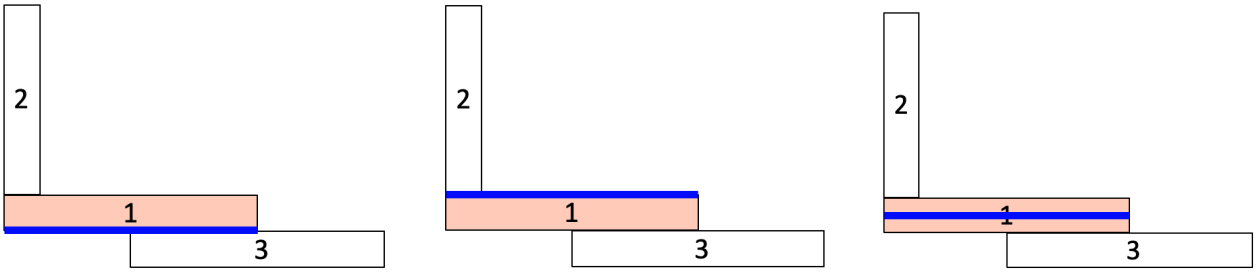
The objective is for the agent to discover a policy π that maps states to actions, with the aim of maximizing long-term rewards (e.g., rewards) [14]. This process unfolds through repeated interactions with the environment. In some scenarios, multiple agents pursue either shared or individual objectives within the environment, a concept known as *Multi-Agent Reinforcement Learning* [15].

During each iteration, the agent selects optimal actions based on its current state and decision policy. Initially, it explores randomly, gradually learning from rewards and forming a policy. A common technique employed is *Explore and Exploit*, which involves initial exploration followed by the exploitation of learned knowledge [14].

The most prevalent RL variant is *Q-learning*, which seeks to sequence actions for the purpose of maximizing rewards [16]. It can be likened to an agent navigating a maze, progressively discovering the optimal path.

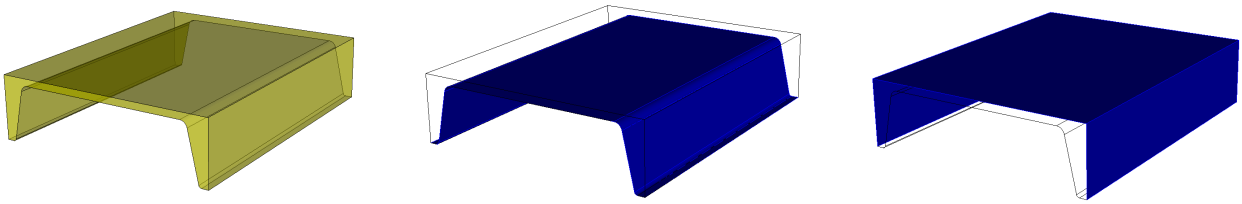
4.2 Overview In our Multi-agent RL framework, we define the initial state space **S** as a network of input volumes, with each volume assigned an agent. These agents are responsible for generating and maintaining a set of actions **A** using commands that convert volumes into 3D manifold representations. For simpler volumes, there are three available actions: **midsurface** and two **copy surface** options (inner and outer planes). Agents employ a decision policy **P** to select a valid **reduce** action. Subsequently, the appropriate commands are executed, and the algorithm uses a reward policy **R** to evaluate the actions, assigning rewards between zero and one.

Rewards are computed based on criteria such as the percentage of maintained connections. For example, if an action causes a volume to lose one of its three neighbors when converted to a sheet body, the agent receives a reward of 0.67. Penalties are applied if small curves or narrow surfaces are generated when



(a) First **Copy Surface** operation (b) Second **Copy Surface** operation (c) **Midsurface** operation

Figure 3: Three primary CAD operations used for reducing a thin volume to a sheet body. Blue curve represents a reduced form of the pink surface



(a) Initial thin volume (b) Reduction solution with 7 surfaces (c) Reduction solution with 3 surfaces

Figure 4: Example of thin volume with multiple surfaces needed to represent its reduction

connecting to neighbor reductions, as small geometries make for less favorable meshing conditions. This process of action-selection and reward-receiving continues until average rewards exceed a user-defined threshold or a maximum iteration count is reached.

Agents enhance action selection by maintaining a history of past rewards for each action. This history informs their choice, favoring more recent rewards through a linearly weighted average mechanism. Agents also incorporate a *randomaction* parameter, allowing them to explore action space by choosing actions randomly instead of relying solely on the highest-scored action. This approach prevents agents from getting stuck in local minima.

Our RL approach is depicted by the flowchart in Figure 5 and is elaborated in the subsequent procedure.

4.3 RL Algorithm

4.3.1 Assumptions: Our RL approach assumes that neighbor volumes are exactly touching and aligned, thus no gaps or overlaps. This ensures that **imprint** and **merge** operations are clean and reliable between volumes. Defeaturing may also be required to remove features like engravings, rounds, chamfers, holes, etc., which wouldn't be appropriately represented in a reduced volume. These assumptions create a conducive framework for the RL process to generate effective reductions.

4.3.2 Input: The input parameters for the RL process include a set of thin CAD volumes, stopping criteria to determine when to terminate the RL iterations, the maximum number of iterations allowed, the learning interval at which training data is saved, and flags for predicting a single iteration or initializing rewards randomly.

- *thin_volumes*: Set of thin CAD volumes adhering to the assumptions mentioned above.
- *stopping_criteria*: RL iterations terminate when the average reward exceeds the specified *stopping_criteria*.
- *max_iters*: Maximum number of RL iterations.
- *learning_interval*: Interval of RL iterations at which training data will be saved.
- *predict*: Boolean flag to execute one iteration of RL and stop.
- *init_random*: Boolean flag to initialize rewards with random numbers.

4.3.3 Output: The output of the RL process is a journal file that contains the sequence of commands

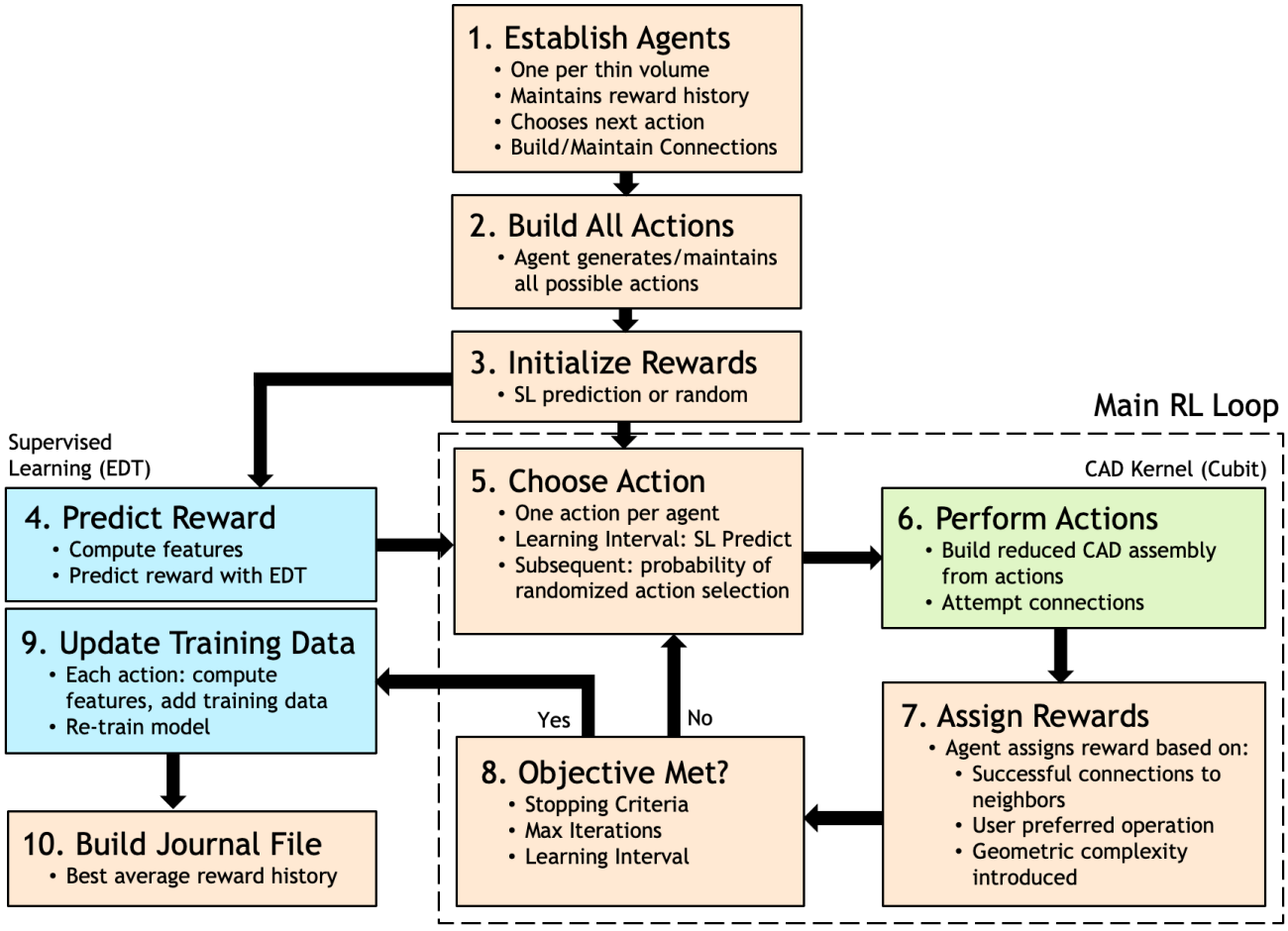


Figure 5: Illustration of the RL Workflow

required to construct the reduced model based on the learned actions and rewards.

- *commands*: Journal file containing ordered commands for building the reduced model.

4.3.4 Method: This method outlines the procedure for applying reinforcement learning to the reduction of thin volumes and the generation of sheet bodies. By establishing individual agents for each volume and employing a graph structure to track their connections, we create a framework for efficient learning. The RL process iteratively selects actions, performs operations in the CAD framework, assigns rewards based on the outcome, and updates the training data. The decision policy allows for exploration and exploitation, ensuring a balance between discovering new actions and leveraging learned knowledge.

1. **Agent Setup:** Each volume in **S** has an agent, managing rewards, action choices, and neighbor interactions. A graph structure records connections through **imprint** and **merge** actions, temporarily connecting 3D thin volumes at merged surfaces to outline anticipated sheet body links. Upon noting connections, the **imprint** and **merge** actions are reversed to return the model to its original state.
2. **Build All Actions:** We gather potential actions into action set **A** for thin volume reduction, encompassing **reduce thin copy** or **midsurface** commands (detailed in Section 4.4). Initial screening excludes actions with known infeasible or suboptimal outcomes.
3. **Initialize Rewards:** Agents start by assigning initial rewards to their actions, ranging from zero to one. A value of zero indicates expected infeasibility, while one represents the most favorable action. These rewards can be predicted using our supervised learning model (Section 3), or random values can be utilized for exploration during the

learning process. Leveraging the previously trained SL model can expedite learning by benefiting from encountered configurations. Ideally, the SL model correctly predicts the configuration if it has been adequately trained. In contrast, employing random initialization disregards prior knowledge and starts with a clean slate.

4. **Predict Rewards:** Rewards can be predicted using our supervised learning EDT model, where computed features for the reduction operation (**copy** or **midsurface**) aid reward prediction. Initial predictions might be less precise for actions with scarce training data or distinct geometry and neighbors. As the RL process advances and gathers more training data, predictions are projected to enhance.

5. **Choose Action:** During every RL iteration, agents opt for an action a from their set of known reduction actions, guided by the decision policy $\mathbf{P}$ explained in Section 5.1. This policy integrates exploration and exploitation, using randomness to explore actions and SL predictions to leverage knowledge from prior iterations.

6. **Perform Actions:** The chosen actions are executed via the CAD kernel. Initially, reduction operations are carried out, succeeded by extending, imprinting, and merging to make the reduced model contiguous.

7. **Assign Rewards:** Agents allocate rewards to their chosen actions using the reward policy $\mathbf{R}$ described in Section 6. These rewards are cataloged in the agents' histories.

8. **Objective Met?** The RL process continues until at least one of the following stopping criteria is met:

   (a) $t_c >$ max_iters, where t_c is the current iteration and max_iters is a user defined maximum number of iterations.

   (b) $R_{t_c} \geq$ stopping_criteria, where $R_{t_c} = \frac{\sum r_a}{N_a}$ represents the average agent reward r_a at iteration t_c for N_a agents.

   (c) User aborts the process: At any iteration $t_c <$ max_iters, the user can choose to cancel the learning procedure if they deem the learning to be sufficient.

   If any of the stopping criteria are met, the process proceeds to step 9. Otherwise, t_c is incremented by 1 and the process returns to step 5.

   Learning Interval: We update the training data at intervals defined by t_c mod (learning_interval) = 0. When this holds, we proceed to step 9 for the update and then return to step 5. Typically, a learning_interval of 5 is effective.

9. **Update Training Data:** Rewards r_a assigned to actions by agents are used as ground truth for our supervised learning model. For completed RL, rewards from the iteration with the highest reward t_{best} expand the SL model. During ongoing RL, the SL model updates every *learning_interval* using rewards from iteration t_c. This refresh of training data and new learning model effectively updates decision policy $\mathbf{P}$.

10. **Build Journal File:** Finally, the agent history is used to extract an ordered list of CAD commands, which are then compiled into a journal file based on the iteration t_{best}.

4.4 Building Action Command Solutions To navigate the design space, our RL approach uses CAD-specific commands as implemented in the Cubit® Geometry and Meshing Toolkit [20] to construct reduced sheet representations from 3D volumes. These actions are categorized into two types:

1. **Reduce Actions:** These encompass CAD commands designed to produce a sheet body representation from a single volume, disregarding considerations for neighboring connections. The following are examples of CAD-specific operations that execute reduction actions:

   (a) `reduce volume <ids> thin copy surface <ids> loft factor <value>... thickness <value>... [delete] [preview]`

   (b) `reduce volume <ids> thin midsurface surface <ids> loft factor <value> thickness <value> [delete] [preview]`

   (c) `merge volume volume <ids>`

   In this context, we introduce new CAD operators that extend the traditional geometric operations of **copy** or **midsurface**. These new operators maintain parent-child relationships, ensuring that the parent 3D volume isn't deleted, and thus both the sheet body representation and the original volume persist within the environment. This preservation enables visualization and compatibility checks. Additionally, the operators incorporate a loft factor that specifies the relative position with the original volume and thickness of the shell for simulation.

These values are extracted during the reduction operation and retained within the geometric model. Furthermore, the operators also manage material associations, which are essential for use in simulation tools.

The **merge** operation is included here to combine separate sheet bodies derived from the same parent solid volume. This ensures that if multiple sheet bodies are needed to represent the 3D volume, they will be merged into a single, continuous, and connected manifold representation for the entire volume.

2. **Connect Actions:** These are CAD commands that modify the resulting sheet bodies using the **tweak** functionality to fill gaps, and include **imprint** and **merge** operations to connect neighboring sheet bodies from neighboring 3D thin solid volumes. The following are examples of connect actions:

   (a) `tweak curve <id_list> target surface <id_list> [preview]`

   (b) `tweak curve <id_list> target curve <id_list> [preview]`

   (c) `tweak curve <id> <id> corner [preview]`

   (d) `imprint volume volume <ids>`

   (e) `merge volume volume <ids>`

   We found these to be a minimal set of operations to maintain most connectivity situations between neighboring sheet bodies, however additional operations may be explored to improve robustness and expand application for future work.

The process of generating CAD commands for reducing thin 3D volumes to sheet bodies follows a structured algorithm. Initially, pairs of opposing surfaces on the thin volume are identified, and the distances between them are computed. After eliminating non-overlapping or dissimilar pairs, continuous solutions are sought for complex cases. The algorithm then crafts commands for both continuous surface sets and individual surface pairs, using calculated distances to define thickness. Additionally, **merge** commands are generated for multi-surface **copy** commands, ensuring a unified manifold of surfaces. This systematic approach effectively translates the 3D volume into its reduced sheet body representation.

Conversely, the procedure for generating connection actions involves connecting sheet bodies from neighboring thin volumes. The algorithm encompasses multiple steps, including creating various **tweak** commands to address different connection scenarios. These include *curve to surface* and *curve to curve* tweaks for specific angles and distances, as well as *corner tweaks* for simultaneous surface extensions. **Imprint/merge** commands are produced when sheet bodies are in close proximity. This sequence is repeated for each surface in the sheet body sets, enabling the formation of robust connections. The procedure concludes by removing solutions that aren't relevant to the specified nearby entity.

5 RL Decision Policy

The decision policy (**P**) prescribes how the agents will select from their known valid actions and is represented as step 3 in figure 5.

5.1 Algorithm for Choosing Actions

5.1.1 Input:

- *agent:* Agent represents a single 3D thin volume
- *iteration:* Current RL iteration
- *init_random:* Whether to initialize with random rewards or from SL predictions
- *random_probability:* Probability for selection of random reward. (default 0.2)
- *learning_interval:* SL model is updated at RL iterations evenly divisible by *learning_interval*

5.1.2 Output: A CAD command(s) or *action* reducing a 3D volume to one or more sheet bodies.

5.1.3 Method: The agent selects an action from its list of valid actions (CAD commands) based on one of the following criteria

1. **Initialization/Reinitialization from SL:** if ($t = 0$ and $init\_random = false$) or $iteration$ mod $learning\_interval = 0$, then select the action from the last iteration. This presumes that the training model has been updated after the last iteration.
2. **Initialization from random:** if $t = 0$, $init\_random = true$ and t mod $learning\_interval \neq 0$, select a random action. If total number of actions for this agent is 2, select the other action
3. **Random exploration:** If a random number, ρ $(0 < \rho < 1) \leq random\_probability$ and max action reward, $r_a < 1.0$, select a random action.
4. **Weighted action:** If $\rho > random\_probability$ then select the reward from the agent's recent re-

ward history based on a decaying weighted average, such that more recent iterations will be given preference.

6 RL Reward Policy

The reward policy (**R**) is based on the following rules:

1. **Connectedness (r_c):** An initial r_c is computed based on the number of successful connections to the reduced sheet body solution. If the reduced sheet body representation maintains all the expected connections to adjacent volumes, as its parent 3D volume did, r_c is set to 1.0. For example, if a parent volume is originally connected to three neighboring volumes and its reduced sheet body representation accurately preserves these connections, r_c is 1.0. If only a subset of the expected connections are maintained, r_c is adjusted accordingly. For instance, if 2 out of 3 connections are preserved, r_c is set to 0.666.
2. If the reduced sheet body representation maintains all the expected connections to adjacent volumes, as its parent 3D volume did, rc is set to 1.0.
3. **Small Curves Penalty (p_c):** In addition to the connectedness reward, a penalty is applied if the reduction action introduces small curves. The penalty factor, p_c is set to 0.9 for each additional small curve generated. Small curves are defined as curves with a length smaller than the parent 3D volume thickness. The penalty ensures that actions leading to a reduced model with fewer small curves are favored.
4. **Narrow Surfaces Penalty (p_s):** Similarly, a penalty factor, p_s, of 0.9 is applied if the reduction action generates narrow surfaces. Narrow surfaces are surfaces with a width smaller than the thickness. The penalty encourages actions that result in reduced models with wider surfaces.
5. **Midsurface vs Copy Penalty (p_m):** An additional penalty factor, p_m, of 0.9 is introduced if a **copy** surface operation is used instead of a **midsurface** operation when both options are available. This penalty allows us to prioritize the use of **midsurface** operations if preferred by the user or if it produces more favorable reductions. If the preferred **midsurface** operation is used or if no **midsurface** operation is available, p_m is set to 1.0.

The final reward, r_a, assigned by the agent for its action is calculated as $r_a = r_c \times p_c^{NSC} \times p_s^{NNS} \times p_m$, where NSC represents the number of small curves generated and NNS represents the number of narrow surfaces generated. This reward policy ensures that actions leading to reduced models with better connectedness, fewer small curves, wider surfaces, and preferred reduction methods are favored and receive higher rewards.

7 Implementation

The implementation encompasses command-line and GUI options, along with accessibility via a Python interface. Users can operate in two distinct modes:

1. **Learn Mode:** This mode initiates an RL loop and executes a specified number of iterations. RL serves as a mechanism to construct an SL model, with RL iteration-produced rewards serving as labels. Learning continues until predefined stopping criteria are met, and a Journal file records the executed commands.
2. **Predict Mode:** In this mode, the SL model is exclusively executed, ensuring a result is always generated. However, the effectiveness and accuracy of the outcome are significantly improved when prior learning has been applied to a similar model. The Journal file is generated solely based on SL predictions.

In practice, for a model consisting of approximately 100 volumes, *Learn* Mode can produce ample training data for the SL model in just a few hours, while *Predict* Mode typically runs in less than a minute on a similar model. Although these tools greatly enhance efficiency, users might encounter connection issues in complex assemblies. Fortunately, the Journal file pinpoints areas of connection failure, enabling users to swiftly detect and resolve issues. Furthermore, interactive tools are available to visualize and promptly address any unresolved issues.

8 Heuristic Approach

Alongside the development of our reinforcement learning (RL) approach for thin volume reduction, we crafted a traditional methodology using heuristic geometric reasoning methods that do not employ machine learning. This alternative method aims to identify individual reduction solutions interconnected to neighboring volumes within the assembly. It achieves this by establishing a graph-like representation of the assembly for efficient traversal. Subsequently, it endeavors to find the longest connected path of reductions while minimizing the creation of sliver surfaces. The algorithm's key steps include:

1. Initialize all possible **copy** reductions for each thin volume.

2. Perform **imprint** and **merge** operations to create a non-manifold, contiguous model of the thin volumes.
3. Reassemble reduction solutions for each volume, considering surfaces split during **imprint** and **merge**.
4. Identify mandatory reductions where volumes share common surfaces with other reductions.
5. Check for infeasible reductions (see figure 32) and terminate if present.
6. Traverse from mandatory reductions, finding the longest connected path of reductions.
7. Prioritize reductions that align with common curves at connections to avoid thin surface creation.
8. Repeat steps 6 and 7 for remaining reductions until all thin volumes are reduced.
9. Identify and extend curves in reductions to maintain connections with adjacent reductions.
10. **Imprint** and **merge** all reductions to connect sheet bodies.

This alternative approach complements RL, offering fresh insights and solutions, aiding in RL policy identification. It played a vital role in our analysis to assess RL's advantages over traditional geometric methods without RL.

9 Examples

The reinforcement learning methods are showcased through a series of practical examples, utilizing CAD assembly models primarily sourced from GrabCAD [17], a widely-used online CAD parts repository. Each example consists of assemblies comprised of multiple intricate parts, with names corresponding to those found on GrabCAD for easy reference shown in figures 6 to 10.

Following the defeaturing process, the reinforcement learning algorithm is applied, with execution times varying based on problem complexity, ranging from minutes for simpler cases to hours for more intricate ones. A learning interval of 5 is employed in these examples, updating the supervised learning model every 5 iterations and reinitializing actions with the newly learned ones.

Table 1 summarizes reinforcement learning outcomes for each test case, accompanied by sheet body model images for reference. Figures 11 to 30 display the defeatured version of the CAD model used as input to the RL method, followed by the resulting sheet body representation produced. Close up versions of the CAD model and sheet body models are also provided.

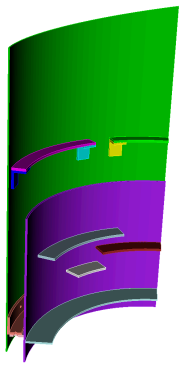


Figure 11: quarter-cannister CAD Model with 13 volumes

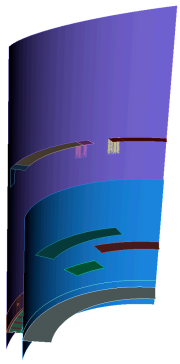


Figure 12: quarter-cannister result sheet bodies

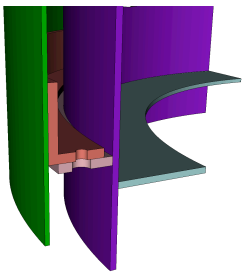


Figure 13: Close Up quarter-cannister 3D volumes

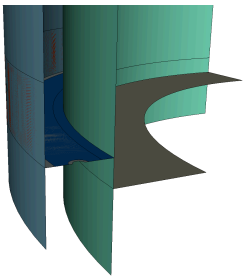


Figure 14: Close Up quarter-cannister sheet bodies

9.1 Test Cases Table 1 presents results for the 5 illustrated examples (Figures 6 to 10). The table includes key measures that offer insights into the performance and characteristics of the reinforcement learning and heuristic methods.

- **Volumes:** The number of volumes in the initial defeatured assembly used for RL, reflecting assembly complexity.
- **Surfaces:** The number of surfaces in the initial defeatured assembly used for RL, representing surface intricacy.
- **Total Connections:** The count of connections between solid volumes determined by geometric proximity, akin to graph edges with volumes as nodes.
- **Max Reward:** Indicates the highest reward achieved in RL iterations. A value of 1.0 represents

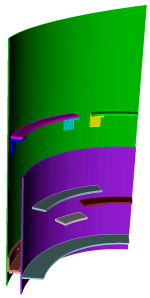


Figure 6: quarter-cannister

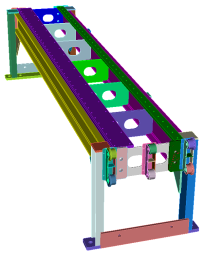


Figure 7: cnc-tube-cutter-1

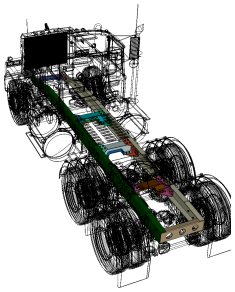


Figure 8: mack-superliner-3

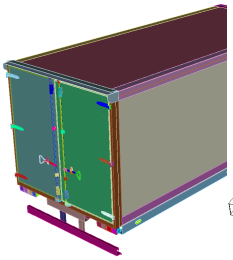


Figure 9: BAU_VUC

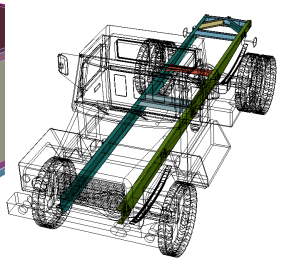


Figure 10: gaz-33086-1

Table 1: Test Results. See section 9.1 for full description of data.

	quarter-cannister	cnc-tube-cutter-1	mac-superliner-3	BAU_VUC	gaz-33086-1
Volumes	13	33	30	80	7
Surfaces	99	298	357	916	128
Total Connections	15	62	55	171	11
Max. Reward	1.00	0.970	0.943	0.704	0.686
Conns. Succeeded	15	62	53	124	6
Conns. Failed	0	0	2	47	5
RL Sheet Bodies	17	41	90	183	25
Heuristic Sheet Bodies	13	33	30	failed	7
RL Merged Curves	42	141	248	408	44
Heuristic Merge Curves	42	138	157	failed	34

perfect performance, while 0.0 indicates no criteria were met.

- **Connections Succeeded:** The number of successful connections established after reduction, showing the effectiveness of the reduction process.
- **Connections Failed:** The count of unresolved connections after reduction, highlighting potential design or connectivity issues.
- **RL Sheet Bodies:** Represents the number of sheet bodies resulting from the RL method's iteration with the highest reward, evaluating its effectiveness in generating sheet bodies.
- **Heuristic Sheet Bodies:** Quantifies the sheet bodies created using the heuristic method on the same defeatured parts, enabling a comparative evaluation against the RL method.
- **RL Merged Curves:** Signifies the count of curves merged within the sheet body model using the RL method to establish connections, reflecting the method's efficiency in forming connections.
- **Heuristic Merge Curves:** Quantifies the merged curves in the resulting sheet body model from the heuristic method, providing insights into the RL method's performance compared to the heuristic approach.

In our exploration of reinforcement learning (RL) for thin volume reduction, we began with the quarter-cannister example, achieving a maximum reward of 1.0, although in practical cases, such high rewards are challenging and require preparatory defeaturing work. Notably, the presence of features like fillets and chamfers can hinder connections, suggesting the need for feature removal actions in RL. Complex T-configurations posed challenges, advocating for their division into multiple volumes as part of RL actions. While reduction was generally successful, connection failures often occurred, necessitating additional operations like tweaking and extending. Identifying various cases for extension and tweaking is vital, prompting further research in this direction. Overall, our observations underscore RL's potential in enhancing thin volume reduction, with room for improvement through defeaturing and handling complex configurations.

In comparing the heuristic approach with RL, the number of merged curves is a key metric. RL consistently merges more curves than the heuristic method,

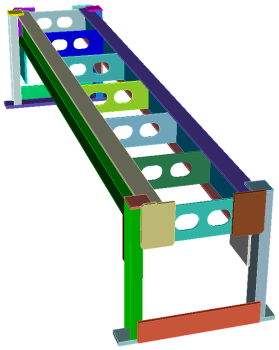


Figure 15: Defeatured: cnc-tube-cutter-1

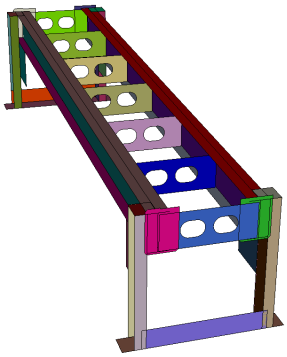


Figure 16: Result Sheet Bodies: cnc-tube-cutter-1

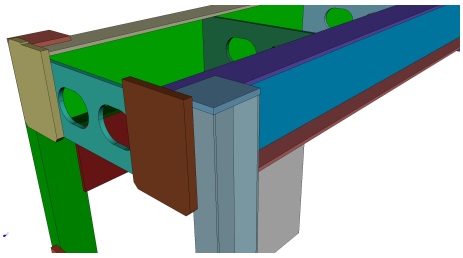


Figure 17: Close Up CAD Model: cnc-tube-cutter-1

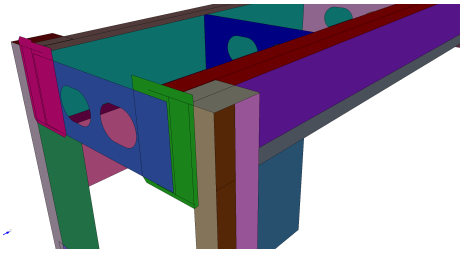


Figure 18: Close Up Sheet Bodies: cnc-tube-cutter-1

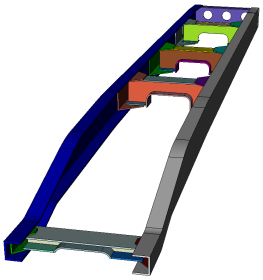


Figure 19: Defeatured: mack-superliner-3

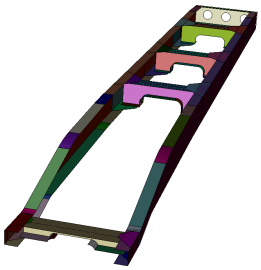


Figure 20: Result Sheet Bodies: mack-superliner-3

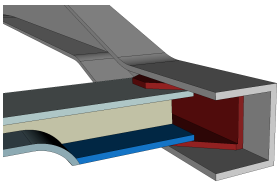


Figure 21: Close Up CAD Model: mack-superliner-3

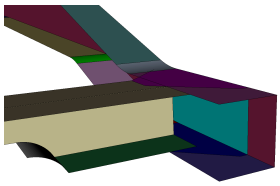


Figure 22: Close Up Sheet Bodies: mack-superliner-3

indicating better performance. In the BAU_VUC test, the heuristic method failed due to an infeasible condition illustrated figures 31 and 32. On the other hand, RL handles such conditions, alerts the user, and provides results for feasible connections. On untrained models, when it is successful, the heuristic method is more efficient than RL. However, after training, their performance is similar.

The current heuristic method is limited to the **copy** operation and excludes **midsurface**, simplifying its implementation. While this is an implementation constraint, it can be expanded for further exploration. Modifying the heuristic's logic, however demands code changes by a developer. In contrast, with RL, users can refine the logic by adding training data. If the RL model falls short, it can be updated with new use cases and rewards. Such adaptability isn't feasible with the heuristic approach.

A distinct advantage of RL is its ability to produce a detailed log of individual commands for each operation. This not only allows users to review and adjust the process but also makes the resulting command logs from RL better suited for archiving and sharing. Such transparency enables others to verify and reuse the procedures. In contrast, the heuristic method only provides a single output without user feedback, making RL more suitable for larger, more complex models.

Table 2 displays the results from RL-generated journal files for each test case, detailing key command operations like **reduce**, **merge**, **imprint**, and **tweak**. The table emphasizes the command frequency in each sequence, shedding light on the automation level RL offers versus manual methods. It underscores the complexity of procedures analysts use for shell model preparation and highlights the potential time savings, reducing what could be days or weeks of manual effort.

Our method efficiently converts thin volume assemblies into interconnected sheets but faces issues with configurations involving three or more stacked volumes, as depicted in Figures 31 and 32. Table 1 highlights these frequent infeasibility issues. Potential solutions could include modifying surface loft properties or employing beams as fasteners to connect coplanar surfaces

Table 2: Counts of Cubit® commands generated by the proposed RL method for each test case. Indicates the relative complexity of procedures generated with RL.

	quarter-cannster	cnc-tube-cutter-1	mac-superlinr-3	BAU_VUC	gaz-33086-1
Reduce	13	33	30	80	7
Merge	19	65	69	244	17
Imprint	15	62	55	171	11
Tweak	5	13	19	30	3
Total	52	173	173	525	38

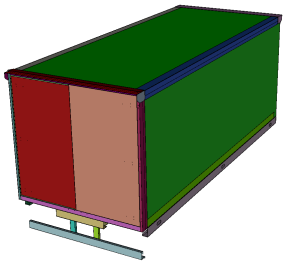


Figure 23: Defeatured: BAU_VUC

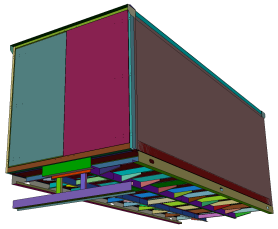


Figure 24: Result Sheet Bodies: BAU_VUC

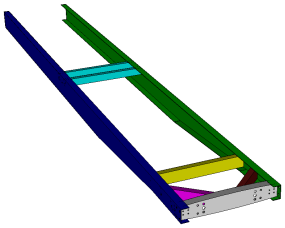


Figure 27: Defeatured: gaz-33086-1

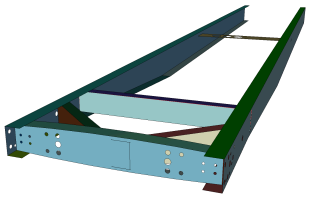


Figure 28: Result Sheet Bodies: gaz-33086-1

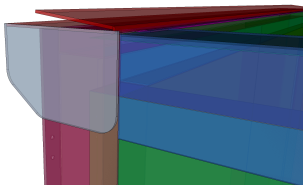


Figure 25: Close Up CAD Model: BAU_VUC

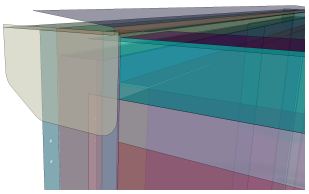


Figure 26: Close Up Sheet Bodies: BAU_VUC

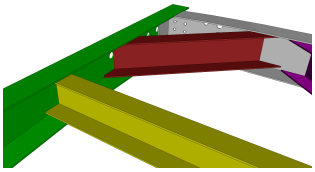


Figure 29: Close Up CAD Model: gaz-33086-1

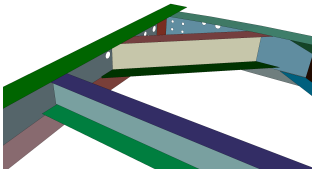


Figure 30: Close Up Sheet Bodies: gaz-33086-1

separated by a thickness. Continued research aims to enhance our approach for such intricate assemblies.

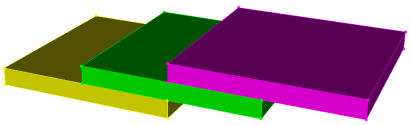


Figure 31: Example of infeasible condition

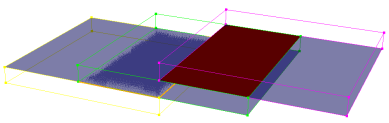


Figure 32: Preview of sheets at infeasible condition

10 Conclusion

In this paper, we have explored the application of reinforcement learning (RL) techniques for thin volume reduction in complex CAD models, revealing their potential to automate and optimize the transformation of 3D solid volumes into sheet bodies while preserving model integrity. Through extensive experiments, we have demonstrated RL's effectiveness in generating reduction solutions across various test cases with minimal manual intervention, thanks to its adaptive learning capabilities. We've also identified key challenges, emphasizing the importance of integrating defeaturing and addressing complex configurations within RL. A comparison with traditional heuristic methods highlighted RL's superiority in establishing connections, as well as its ability to handle undefined conditions and provide user notifications. This research sets the stage for advancements in thin volume reduction, offering significant time savings in CAD modeling and enhancing productivity across industries.

References

[1] D. Chapelle and K-J. Bathe, *The Finite Element Analysis of Shells - Fundamentals Computational Fluid and Solid Mechanics*, Springer-Verlag, New York, 2011.

[2] B. N. Maker, *NIKE3D: A nonlinear, implicit, three-dimensional finite element code for solid and structural mechanics*, [Online]. Available: `https://www.osti.gov/biblio/6112915`, 1991.

[3] O. C. Zienkiewicz, *The finite element method, Fifth edition Volume 2: Solid Mechanics*, Butterworth Heinemann, Oxford, 2002.

[4] CadFix: Medial Object Technology, Available online: `https://www.cadinterop.com/en/your-needs/cad-data-reuse-for-cae/medial-object-technology.html`, Accessed: 2023-09-14.

[5] I.H. Choi, C.S. Woo, H.P. Moon, *Mathematical Theory of Medial Axis Transform*, *Pacific Journal of Mathematics*, vol. 181, no. 1, 1997.

[6] C. G. Armstrong, *Modelling Requirements for Finite-Element Analysis*, In *Computer-Aided Design*, Volume 26, Issue 7, July 1994, Pages 573-578.

[7] K. Suresh, *Automating the CAD/CAE Dimensional*, In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, June 2003, pp. 7685.

[8] L. Sun, C. M. Tierney, C. G. Armstrong, T. T. Robinson, *Automatic Decomposition of Complex thin Walled CAD Models for Hexahedral Dominant Meshing*, In *Procedia Engineering*, Volume 163, Pages 225-237, 25th International Meshing Roundtable, 2016.

[9] Ansys SpaceClaim, *Beam and Shell Modeling*, [Online]. Available: `https://www.ansys.com/products/3d-design/ansys-spaceclaim`, Accessed: 2023-09-14.

[10] A. Purwar, K. A. Desai, S. Canfield, R. Rai, Z. Nie. "Special Issue: Machine Learning and Representation Issues in CAD/CAM." *ASME. J. Comput. Inf. Sci. Eng.* January 2024; 24(1): 010301. `https://doi.org/10.1115/1.4064059`

[11] S. J. Owen, T. M. Shead, and S. Martin, *CAD Defeaturing Using Machine Learning*, 28th International Meshing Roundtable, Buffalo, New York, USA, February 6, 2020. DOI: `https://doi.org/10.5281/zenodo.3653426`

[12] S. J. Owen, A. Carbajal, M. Peterson, and C. Ernst, *Machine Learning Classification and Reduction of CAD Parts for Rapid Design to Simulation*, SIAM International Meshing Roundtable, March 6-9, 2023, in Amsterdam, Netherlands

[13] Martin J. Osborne and Ariel Rubinstein, "An Introduction to Game Theory", 2004, MIT Press.

[14] L. Pack Kaelbling, M. L. Littman, and A. W. Moore, *Reinforcement learning: A survey*, Journal of artificial intelligence research, vol. 4, pp. 237–285, 1996.

[15] Changxi Zhu, Mehdi Dastani, Shihan Wang, "A Survey of Multi-Agent Reinforcement Learning with Communication", 2022, arXiv:2203.08975 [cs.MA].

[16] Francisco S. Melo, *Convergence of Q-learning: A simple proof*, Institute Of Systems and Robotics, Tech. Rep, pp. 1-4, 2001.

[17] GrabCAD, *Making Additive Manufacturing at Scale Possible*, [Online]. Available: `https://grabcad.com`, Accessed: 2024-01-15.

[18] L. Breiman, *Random forests*, *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001, Springer.

[19] F. Pedregosa, et. al. "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[20] Sandia National Laboratories. The Cubit® Geometry and Mesh Generation Toolkit. `https://cubit.sandia.gov`. Accessed: 2024-01-15.

[21] Dassault Systmes. 3D Acis modeler. `https://www.spatial.com/products/3d-acis-modeling`. Accessed: 2024-01-15