FeatureFox: Sample-Efficient Panoptic Graph Segmentation for Machining Feature Recognition in B-Rep 3D-CAD Models

Bertram Fuchs[1,2] Altay Kacan[2,3] Aaron Haag[2] Oliver Lohse[2]
[1]Technische Universität Berlin [2]Siemens AG, München, Germany
[3]Friedrich-Alexander-Universität Erlangen-Nürnberg
{bertram.fuchs, altay.kacan, aaron.haag, oliver.lohse}@siemens.com

***Abstract*—Automatic feature recognition (AFR) on B-Rep 3D-CAD models is central to CAD/CAM automation, yet most learning-based methods are complex, data-hungry, and evaluate instance grouping and semantic labeling separately. We present FeatureFox, a *panoptic AFR* pipeline that outputs machining *instances with semantic labels*: a calibrated binary edge classifier on enriched edge attributes localizes feature boundaries, instances are recovered as connected components in a pruned face-adjacency graph, and a per-instance classifier predicts the machining class from aggregated subgraph attributes. We evaluate on MFInstSeg using Panoptic Quality (PQ), which jointly scores instance separation and semantic correctness. FeatureFox is substantially more sample- and compute-efficient than the deep baseline AAGNet, reaching $\mathrm{PQ} > 0.9$ with ~ 250 training parts versus $\sim 5{,}000$ for AAGNet, and training on the full MFInstSeg set takes seconds on a GPU. On the full training set, AAGNet surpasses FeatureFox marginally in PQ, while FeatureFox remains slightly ahead in feature-level recognition and localization accuracy. Finally, leveraging its low data requirement, we train FeatureFox on 270 manually labeled industrial CAD parts and show qualitative generalization to an unseen real industrial part, indicating practical real-world applicability.**

***Index Terms*—automatic feature recognition (AFR), B-Rep 3D-CAD, face-adjacency graph, panoptic graph segmentation, calibrated classification**

I. Introduction

Computer-aided process planning (CAPP) connects design to manufacturing by translating CAD models into CAM decisions such as setups, toolpaths, and operation sequences. One key bottleneck in this pipeline is automatic feature recognition (AFR): identifying manufacturable machining feature instances from CAD geometry and topology. When AFR is unreliable, downstream planning can become error-prone and may require manual intervention, limiting CAD/CAM automation.

AFR is challenging because real mechanical parts rarely present features in clean isolation. Features intersect, blend, and appear partially due to design variations or modeling conventions, and small geometric edits can significantly alter local topology. In addition, process planning requires more than semantic category labels: it requires separate feature instances with consistent boundaries. These requirements naturally align with the task of panoptic segmentation, where semantic and instance information must be inferred jointly.

Classic AFR approaches are knowledge-driven (rule-based systems, graph matching, volume decomposition, heuristics) and remain attractive due to interpretability [14], [15], but they are not robust to feature intersections and design variability and are costly to maintain when part complexity increases [1], [15]. Learning-based methods improve generalization and achieve strong results on B-Rep representations, yet state-of-the-art models are often complex, data-hungry, and typically evaluate instance grouping and semantic labeling as separate tasks. This creates a practical gap: labeled industrial CAD data is scarce and expensive, so methods that require thousands of annotated parts are not deployable for AFR in industry.

Contributions. To address this, we present FeatureFox and make the following contributions:

- **Panoptic AFR formulation and metric.** We cast machining feature recognition in 3D-CAD on B-Rep faces as a *Panoptic Segmentation* problem (faces per instance + semantic instance label) and advocate *Panoptic Quality (PQ)* as a single metric that jointly evaluates machining feature localization and class recognition.
- **FeatureFox: interpretable hierarchical pipeline.** We introduce FeatureFox, a lightweight tree-based panoptic AFR pipeline that (i) localizes feature instances via a calibrated binary edge-boundary model on enriched edge attributes, (ii) recovers instances as connected components in a pruned face-adjacency graph, and (iii) assigns one semantic label per *whole instance* using aggregated subgraph attributes.
- **Strong sample- and compute-efficiency on MFInstSeg.** On AAGNet's MFInstSeg dataset, FeatureFox reaches $\mathrm{PQ} > 0.9$ with ~ 250 training parts, whereas the deep baseline AAGNet requires $\sim 5{,}000$; training FeatureFox on the full MFInstSeg training set takes seconds on a GPU, potentially enabling offline customer-centric deployment.
- **Practical low-data industrial applicability.** Enabled by its low data requirement, we train FeatureFox on ~ 270 manually labeled real-world CAD parts and demonstrate qualitative generalization to an unseen industrial part, indicating feasibility for real deployment scenarios with scarce annotations.

Due to internal company policy and confidentiality constraints, we cannot publicly release the code and data for this work.

II. Related Work

AFR has been researched since the 1980s [13] as one of the core problems to be solved in CAPP and broadly falls into rule-based and learning-based methods. Rule-based systems recognize machining features via handcrafted geometrical and topological rules or graph-based matching procedures, and remain attractive due to interpretability, but they often break on intersecting or topologically complex features and require substantial engineering effort to maintain at scale [1], [9], [12].

Learning-based AFR progressed mainly by changing the input representation and moved toward CAD-native models. Early deep methods used voxels [19], multi-view renders [10], [11] or point-clouds [6], but these representations either lose topological information or are computationally heavy and depend on view selection [2], [11]. More recent work operates directly on B-Rep–derived graphs [2] and combines geometric UV-grid sampling with message passing in a GNN [4]. These models typically predict per-face semantic labels, i.e., semantic segmentation on B-Reps.

Only a few methods explicitly address instance-aware AFR, separating distinct feature instances in addition to semantic labeling. ASIN jointly clusters instances and predicts semantics from point clouds [18], while AAGNet introduces a B-Rep–native gAAG representation with a multi-head CNN+GNN pipeline for semantic and instance predictions [16]. MFTReNet follows AAGNet's gAAG-based multi-task GNN design, but extends it with additional heads to predict inter-feature topological relations [17]. EAGIS targets instance grouping via edge relations on MFInstSeg but does not address semantic labeling [7]. Overall, most prior learning-based methods either focus on semantic segmentation or instance segmentation only or do not provide a single unified panoptic output. To the best of our knowledge, all current State-of-the-Art learning-based AFR methods require thousands of labeled CAD parts and have not demonstrated training and evaluation on real industrial CAD files, limiting industrial applicability because annotations are scarce and costly.

III. Methodology

Previous methods that perform well on the task of AFR all train a hybrid model consisting of a CNN and a Graph Neural Network [16], [17]. We define the task as a Panoptic Segmentation [5] task on Graphs and propose FeatureFox, a hierarchical calibrated Tree-based model that recognizes feature instances and their respective semantic class.

A. Graph representation of B-Reps and features

Modern CAD systems usually represent the topology and geometry of CAD parts using Boundary Representation (B-Rep). A B-Rep describes a solid by its enclosing surfaces, edges, and vertices, and their connectivity (topology) and geometry. As B-Rep is a variable-sized, CAD-kernel dependent geometric and topological data structure and not a fixed-dimensional numeric representation suitable for Machine Learning, a representation format called Geometric Attributed Adjacency Graph (gAAG) [16] was proposed. EAGIS changed the representation format and used edges as nodes for training a GNN, and formulated the problem of feature instance detection as binary edge prediction task [7]. FeatureFox builds on the gAAG representation, and leverages face adjacency to propagate neighboring face information to edges.

a) Face-adjacency graph.: Given a B-Rep with faces $\mathcal{F} = \{f_1, \dots, f_n\}$ and B-Rep edges $\mathcal{E} = \{e_1, \dots, e_m\}$, let $\mathrm{Inc} : \mathcal{E} \to 2^{\mathcal{F}}$ map each B-Rep edge to its incident faces. Assuming a 2-manifold B-Rep, each edge is shared by exactly two faces:

$$\forall e \in \mathcal{E} : \quad \mathrm{Inc}(e) = \{f_i, f_j\}, \;\; i \neq j. \tag{1}$$

We define the undirected face-adjacency graph $G = (V, E)$ with one node per face,

$$V := \mathcal{F}, \tag{2}$$

and an (undirected) graph edge between two nodes iff the corresponding faces share a B-Rep edge:

$$E := \big\{\{f_i, f_j\} \subseteq \mathcal{F} \;:\; \exists e \in \mathcal{E} \text{ s.t. } \mathrm{Inc}(e) = \{f_i, f_j\}\big\}. \tag{3}$$

Equivalently, the adjacency matrix $A \in \{0,1\}^{n \times n}$ is

$$A_{ij} = \begin{cases} 1, & \exists e \in \mathcal{E} : \mathrm{Inc}(e) = \{f_i, f_j\}, \; i \neq j, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

b) Per-face and per-edge attributes.: Let the B-Rep contain $n = |\mathcal{F}|$ faces and $m = |\mathcal{E}|$ (manifold) edges, and let $\mathrm{Inc} : \mathcal{E} \to 2^{\mathcal{F}}$ map each B-Rep edge to its incident faces with $\mathrm{Inc}(e_k) = \{f_{s(k)}, f_{t(k)}\}$, where $s, t : \{1, \dots, m\} \to \{1, \dots, n\}$ assign an arbitrary (but fixed) ordering of the two incident face indices of edge e_k. We associate one attribute vector to each face and to each B-Rep edge, where d_F is the dimensionality of the face attribute vector:

$$\mathbf{x}_i^F \in \mathbb{R}^{d_F} \quad (i = 1, \dots, n), \qquad \mathbf{x}_k^E \in \mathbb{R}^{d_E} \quad (k = 1, \dots, m), \tag{5}$$

.

c) Face UV-grid tensor.: For each face f_i we sample a regular grid like in UV-Net [4] in its parametric domain with resolution $U \times V$. For grid indices (u, v) we store 3D position $\mathbf{p}_i(u, v) \in \mathbb{R}^3$, unit normal $\mathbf{n}_i(u, v) \in \mathbb{R}^3$, and a binary validity mask $\mu_i(u, v) \in \{0, 1\}$ indicating whether the sample lies on the trimmed face. We stack these three in the face-grid tensor $\mathbf{T}^F \in \mathbb{R}^{n \times U \times V \times 7}$.

d) Edge sampling tensor.: Analogously, we sample each B-Rep edge e_k with M points along its parameter and store a 15-dimensional geometric descriptor per sample. This yields the edge tensor $\mathbf{T}^E \in \mathbb{R}^{m \times M \times 15}$ that stacks for each edge sample the 3D point $\mathbf{p}_k(j) \in \mathbb{R}^3$, the unit tangent $\mathbf{t}_k(j) \in \mathbb{R}^3$, the unit normal $\mathbf{n}_k(j) \in \mathbb{R}^3$, the unit binormal $\mathbf{b}_k(j) \in \mathbb{R}^3$, $\mu_k(j) \in \{0, 1\}$ a trim mask indicating whether the sample lies inside or on the valid edge, $r_k(j) \in \mathbb{R}$ the (local) radius, and $\tau_k(j) \in \mathbb{R}$ the torsion.

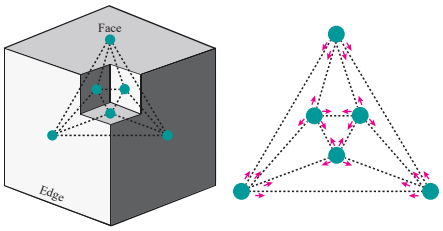

Fig. 1. Graphical representation and information propagation from vertices (B-Rep faces) to edges (B-Rep edges). Face Adjacency Graph as used in UV-Net [4] (Left). Edge samples and per-edge attributes are used to calculate the per-edge attribute feature vectors. Per-face attributes and additional attributes calculated from per-face samples are propagated towards the edges and enrich the edge attributes (Right).

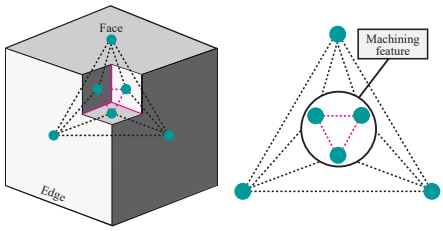

Fig. 2. FeatureFox binary machining feature instance labels. We are labeling according to the question: Are both faces adjacent to the edge in the same machining feature instance? Edges representing machining feature boundaries are labeled with False (black), edges within one feature instance are labeled with True (pink).

B. Instance Segmentation Model

For training our instance model, we first enrich our edge attributes with additional information from neighboring nodes and then train a calibrated binary feature boundary classifier.

a) Enriched edge attributes: For each graph edge $k \in E$ (corresponding to a B-Rep edge incident to exactly two faces), let i and j denote the indices of its two incident faces in the face-adjacency graph; we refer to this edge by the unordered pair $\{i, j\}$. We define additional hand-crafted edge features by combining information from the observed edge and its two neighboring faces, and concatenate these features to the pre-extracted edge attribute vector $\mathbf{x}_k^E$. Concretely,

$$\tilde{\mathbf{x}}_k^E \;=\; \big[\, \mathbf{x}_k^E \,\|\, \Delta\mathbf{x}_k^E \,\big],$$

where $[\cdot\|\cdot]$ denotes concatenation. The information flow is depicted in Figure 1.

Edge attributes include dihedral, concavity, face-normal, coplanarity, size-ratio, centroid-distance, shared-boundary, topological, and edge-type cues, summarizing local geometric and topological information.

b) Labeling paradigm: We formulate feature instance localization as a binary edge-classification problem. For each graph edge $k \in E$ connecting two adjacent faces with indices i and j, let $\ell(i) \in \{1, \ldots, L\}$ denote the ground-truth machining feature instance ID of face i, where L is the number of feature instances. We then define the binary edge label

$$y_k \;=\; [\ell(i) = \ell(j)] \;\in\; \{0, 1\},$$

i.e., an edge is labeled positive iff both incident faces belong to the same machining feature instance. Figure 2 illustrates this labeling: $y_k = 1$ (True, pink) if $\ell(i) = \ell(j)$, and $y_k = 0$ (False, black) otherwise.

c) Per-Edge Calibrated Feature Instance Segmentation: We train a calibrated binary `XGBoost` classifier with 200 trees (maximum depth 6, learning rate 0.1) on the enriched edge features $\tilde{\mathbf{x}}_k^E$ to predict the probability $p(y_k = 1 \mid \tilde{\mathbf{x}}_k^E)$ that the two incident faces of edge k belong to the same machining feature instance. Since our instance localization relies on probability thresholding to decide whether an edge should connect faces within the same instance, well-calibrated probabilities are crucial for a stable and interpretable decision threshold. We therefore apply isotonic calibration using three-fold cross-validation, which is sufficient in our setting because the calibration folds contain a large number of edge samples as each sample B-Rep contains many edges. Finally, we apply a fixed threshold to the calibrated probabilities to obtain binary feature-boundary predictions. We do not use the MFInstSeg validation set for model selection, since we fix all hyperparameters a priori and do not tune them as the resulting instance grouping performance is already high, as shown in Table I.

We then recover machining feature instances as the connected components of a pruned face-adjacency graph. Specifically, given the original face-adjacency graph G and predicted binary edge labels $\hat{y}_k \in \{0, 1\}$, we retain only edges between faces in one feature, dropping feature boundary edges, and define the pruned graph $\hat{G} = (V, \hat{E})$ with

$$\hat{E} \;=\; \{\, k \in E \mid \hat{y}_k = 1 \,\}.$$

Each machining feature instance is then defined as a connected component of $\hat{G}$, i.e., a maximal set of faces that are mutually reachable via paths consisting only of edges in $\hat{E}$.

d) Per-instance Semantic Segmentation.: AAGNet performs semantic classification on a per-face level [16]. In contrast, we classify each *feature instance* as a whole. This choice makes the pipeline explicitly panoptic: instance grouping is performed first, and semantics are then assigned per instance. For each ground-truth feature instance in G, represented by a connected *node set* $C \subseteq V$ (and its induced subgraph $G[C]$), we compute a single instance attribute vector $\mathbf{z}_C$ by aggregating the face sample, edge sample as well as face attribute and edge attribute vectors of all faces and edges contained in C, which can be compared to a manual form of pooling. Feature instance attributes include the number of faces and edges, the log characteristic length (used as scale descriptor)

$$L_{\log} = \log(L + \varepsilon)\,, \qquad L = \sqrt{\sum_{v \in C} A_v},$$

where C denotes the set of faces in the feature instance, A_v is the area of face v, and ε is a small constant for numerical stability, as well as a histogram of face and edge types, curvature distributions, graph topology features (e.g., spectral connectivity), and boundary-structure cues such as the number

TABLE I
PER-FACE PERFORMANCE ON MFINSTSEG TEST SET.

Network	Semantic Segmentation		Instance Grouping		Number of parameters ↓
	Acc(%) ↑	mIOU(%) ↑	Acc(%) ↑	F_1 Score ↑	
AAGNet	99.15 ±0.03	**98.45** ±0.04	99.94 ±0.01	98.84 ±0.07	**0.41 M**
MFTReNet	**99.56** ±0.02	98.43 ±0.03	**99.95** ±0.01	98.90 ±0.02	0.58 M
FeatureFox (Ours)	– –	– –	99.91 ±0.00	**99.45** ±0.00	–

of disconnected boundary-edge loops, among others. We then train a semantic `XGBoost` classifier (400 trees, maximum depth 6) on instance-label pairs $(\mathbf{z}_C, c_C)$, where c_C is the ground-truth semantic label of instance C. As in the edge model, we fix all `XGBoost` hyperparameters a priori and therefore do not perform validation-based model selection. During inference, for each predicted instance subgraph $\hat{C}$ returned by the instance segmentation stage, we compute $\mathbf{z}_{\hat{C}}$ and output its semantic class prediction.

IV. RESULTS

We evaluate on MFInstSeg, introduced by the authors of AAGNet, as it is the most frequently used AFR dataset and includes semantic and instance labels [3], [7], [16], [17].

A. Hardware Setup

For extracting face and edge samples as well as face and edge attributes faster, we convert all $62,477$ STEP files (standardized CAD format) in MFInstSeg to Siemens NX .prt files. For all experiments we used $62,178$ MFInstSeg samples, for which the conversion was successful and no CAD topology or geometry integrity error was detected. For training AAGNet and FeatureFox, we used the same NVIDIA RTX 6000 Ada GPU.

B. Face-level performance on MFInstSeg

In a first ablation study, we evaluate instance segmentation performance of FeatureFox. Table I shows the instance segmentation performance in comparison to AAGNet and MFTReNet [16], [17]. For FeatureFox, we do not state per-face semantic segmentation accuracy, as we train our semantic segmentation model for feature instance subgraphs and not on face-level as explained (see Methods section). FeatureFox performs similarly to AAGNet and MFTReNet for instance segmentation.

C. Feature-level performance on full MFInstSeg training dataset

We compare feature-level recognition and localization performance of AAGNet and FeatureFox, both trained on MFInstSeg. In contrast to prior work that reports instance grouping and semantic labeling separately, we formulate AFR on B-Rep faces as a single panoptic feature-level graph segmentation task: each face is assigned a feature instance ID and a semantic class and the result can be quantified using Panoptic Quality (PQ) as a metric. This unifies instance and semantic segmentation in one output, analogous to panoptic segmentation of image pixels in computer vision.

a) Recognition Localization Accuracy: We first report Recognition Localization Accuracy, defined as the fraction of ground-truth machining feature instances that are fully recovered, meaning that all faces of the instance are assigned to a single predicted feature instance and the semantic class is predicted correctly. Since AAGNet outputs instance grouping and semantic segmentation separately, we convert per-face semantic predictions into per-instance labels. Let $\hat{C} \subseteq V$ be a predicted feature instance, and let $\mathbf{s}_v \in \mathbb{R}^K$ denote the pre-softmax semantic logits for face v over K machining feature classes. We perform logit-sum voting by aggregating

$$\mathbf{S}_{\hat{C}} = \sum_{v \in \hat{C}} \mathbf{s}_v,$$

and assign the instance class as

$$\hat{c}_{\hat{C}} = \arg \max_{k \in \{1,\dots,K\}} \left(\mathbf{S}_{\hat{C}}\right)_k .$$

Equivalently, this corresponds to maximizing the product of per-face class posteriors under a conditional independence assumption.

b) Panoptic Quality (PQ) for machining features.: Again, let ground-truth machining features be instances $\mathcal{G} = \{(C, c)\}$ where $C \subseteq V$ is the set of faces belonging to one feature instance and c is its machining feature class. Similarly, predictions are $\mathcal{P} = \{(\hat{C}, \hat{c})\}$. For a predicted instance $\hat{C}$ and a ground-truth instance C, define the face-based IoU as

$$\mathrm{IoU}(\hat{C}, C) = \frac{|\hat{C} \cap C|}{|\hat{C} \cup C|}.$$

We form a one-to-one matching $\mathrm{TP} \subseteq \mathcal{P} \times \mathcal{G}$ over pairs with $\hat{c} = c$ and $\mathrm{IoU}(\hat{C}, C) > 0.5$ (unmatched predictions are FP and unmatched ground truth are FN). Then

$$\mathrm{PQ} = \frac{\sum_{((\hat{C},\hat{c}),(C,c)) \in \mathrm{TP}} \mathrm{IoU}(\hat{C}, C)}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|} .$$

We believe that PQ is the best metric for evaluating the AFR task as it scores the joint correctness of "what machining feature is it?" and "which exact faces belong to that feature?", in one number. With this metric we couple recognition and localization and penalize merges/splits via FP/FN while rewarding accurate face-level localization via IoU.

For both AAGNet and FeatureFox trained on the full MFInstSeg training set, we report results over three random seeds. For FeatureFox, the `XGBoost` components were effectively seed-insensitive, yielding identical performance across seeds. For AAGNet, hyperparameters were fixed a priori and kept unchanged across all experiments to ensure fair, reproducible comparisons and avoid tuning bias, particularly in the low-data regime. Results are shown on the right side of table II. AAGNet achieves slightly higher PQ results compared to FeatureFox, while FeatureFox has a marginal edge in terms of features recognized completely.

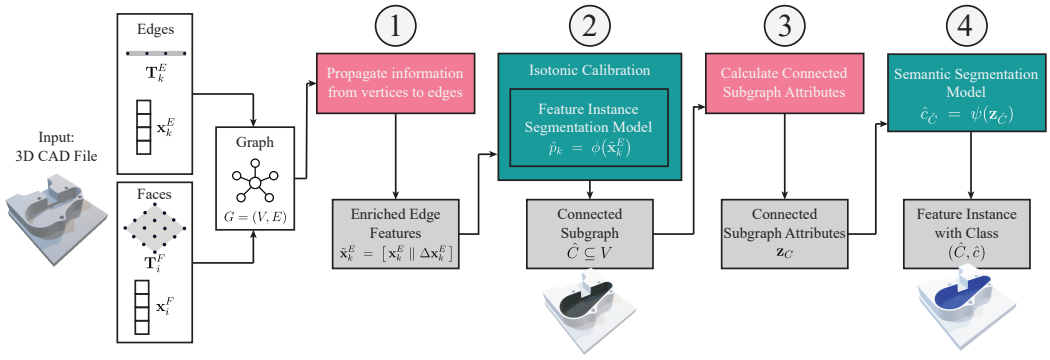

Fig. 3. FeatureFox pipeline. Starting from a B-Rep 3D-CAD model, we construct a face-adjacency graph and attributes, propagate vertex information to edges (1), and localize the feature with a calibrated edge boundary classifier for instance grouping (2). We then compute connected-component attributes (3) and recognize the class of each predicted instance with a semantic model (4). Pink: Attribute calculation. Teal: Model training. Gray: Intermediate representation.

TABLE II
PER-FEATURE PERFORMANCE ON MFINSTSEG AT TWO TRAINING SET SIZES EXCLUDING STOCK FEATURES

Network	1000 training samples		Full training set (43,524 samples)	
	Recognition & Localization Acc(%) ↑	PQ(%) ↑	Recognition & Localization Acc(%) ↑	PQ(%) ↑
AAGNet	65.54 ±2.87	78.81 ±2.28	95.49 ±2.38	**97.44** ±1.16
FeatureFox (Ours)	**93.43** ±0.00	**94.65** ±0.00	**95.99** ±0.00	96.87 ±0.00

D. Sample efficiency on partial MFInstSeg training dataset

FeatureFox is motivated by the scarcity of large, labeled industrial 3D-CAD datasets. With FeatureFox, we therefore design a model that can recognize machining features reliably with limited training parts. To assess sample efficiency, we train AAGNet and FeatureFox on matched training subsets of MFInstSeg with varying sizes (three random seeds each) and evaluate on the full MFInstSeg test set. For AAGNet, we use the default hyperparameters and apply early stopping since convergence requires more epochs when fewer training samples are available. We also experimented with larger learning rates for smaller training subsets but achieved similar results. Figure 4 reports PQ (excluding stock features) for training subsets ranging from 50 to 43.5k samples. FeatureFox consistently performs better in the low-data regime (e.g., 0.83 vs. 0.50 PQ at 50 samples) and remains ahead up to roughly half of the full training set. AAGNet only slightly surpasses FeatureFox on the full dataset (about +1 PQ point). To exceed 90% PQ, AAGNet requires $\sim$ 5,000 training parts, whereas FeatureFox reaches the same level with $\sim$ 250. Table II summarizes results for 1,000 samples and the full training set.

E. Computational efficiency

We trained both FeatureFox and AAGNet on the same NVIDIA RTX 6000 Ada GPU. In our implementation and experimental setup, training AAGNet (excluding data loading) for 100 epochs on the full dataset took 38 minutes with an average GPU utilization of 18%. We did not optimize AAGNet for hardware utilization. Under the same setup, training FeatureFox on the full dataset took 8 seconds.

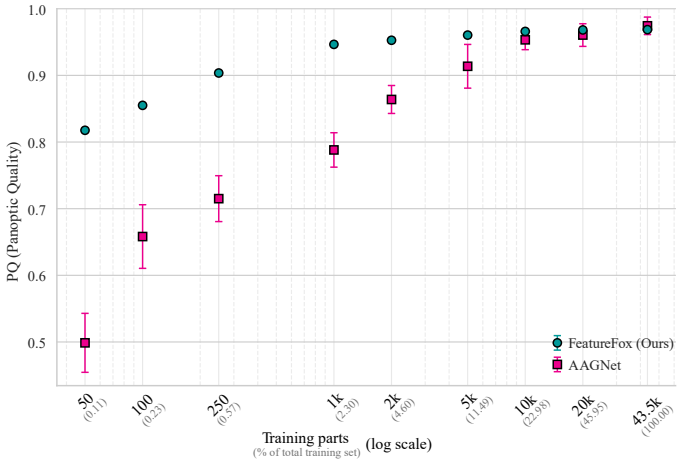

Fig. 4. Sample efficiency on MFInstSeg (excluding stock): PQ versus training-set size for AAGNet and FeatureFox. FeatureFox dominates in the low-data regime and reaches $> 90\%$ PQ with $\sim$ 250 training parts, whereas AAGNet requires $\sim$ 5,000; AAGNet is only marginally better on the full training set.

F. Qualitative transfer to industrial CAD parts

To the best of our knowledge, none of the current state-of-the-art models has been tested on industrial CAD data. The sample efficiency of our model makes it possible to label and train FeatureFox on a small dataset of 270 industrially realistic NX .prt files. We further assess practical feasibility on an unseen publicly available National Institute of Standards and Technology (NIST) 3D-CAD file [8], which was not used for training. The result is shown in Figure 5. Holes, pockets, and slots were mostly recognized correctly in this example. Although minor prediction errors remain, the result suggests qualitative transfer to industrial CAD geometry.

V. DISCUSSION AND LIMITATIONS

FeatureFox shows that panoptic AFR can be achieved with a hierarchical tree-based pipeline: a calibrated binary edge-boundary model yields stable feature instances, and a per-instance classifier assigns semantic labels from aggregated instance attributes. This design excels in the low-data regime, reaching high PQ with a few hundred training parts, whereas

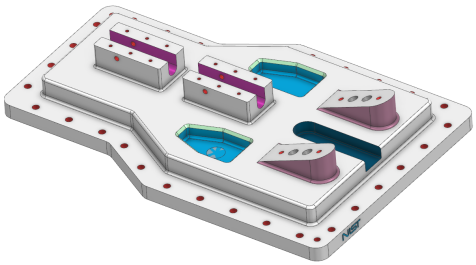

Fig. 5. Inference on an industrial CAD part from the NIST dataset [8]. Most features are recognized correctly. Chamfers were labeled and correctly recognized; edge blends were not labeled. The dark grey multi-step holes on the inclined plane are correctly grouped as instances but misclassified due to very few multi-step-hole training samples. The two pink vertical features around the inclined planes are predicted as freeform but should be stock. The *NIST* logo is segmented into two pocket instances: *N* and *IST* (bottom of figure).

deep AFR architectures typically require thousands. On the full MFInstSeg training set, AAGNet slightly surpasses FeatureFox in PQ, suggesting end-to-end representation learning can exploit subtle geometric cues when sufficient data is available.

A key limitation of FeatureFox is its non-end-to-end nature: boundary errors propagate to semantic labeling. While boundary prediction is strong on MFInstSeg, more ambiguous boundaries would reduce robustness. More broadly, future progress in AFR will likely require labeled industrial CAD data beyond synthetic data, since real parts contain multi-operation and intersecting features whose instances may be disconnected face sets in the B-Rep yet must be treated as a single manufacturable feature, which our method does not handle. Industrial validation is likewise still limited to qualitative results on one unseen public NIST part due to scarce labeled industrial CAD data. Nevertheless, FeatureFox's low compute and data requirements make customer-specific on-premise training and inference feasible, which may be beneficial for privacy-sensitive industrial workflows.

Furthermore, we hypothesize that FeatureFox can generalize to variable-sized CAD graphs at inference time, unlike many GNN-based approaches that are sensitive to shifts in graph structure, as long as edge-boundary and instance-subgraph statistics remain comparable across domains. Moreover, inference is linear in graph size, i.e., $\mathcal{O}(|V| + |E|)$. A systematic cross-distribution evaluation is left for future work.

VI. Conclusion

With FeatureFox, we introduce a panoptic graph segmentation approach for AFR that is competitive with current SOTA methods such as AAGNet [16]. Unlike prior work that evaluates instance grouping and semantic segmentation separately, we frame AFR as a single joint task and advocate Panoptic Quality (PQ) as the most faithful metric for feature-level recognition and localization. On MFInstSeg, FeatureFox requires only a fraction of the training time and is markedly more sample-efficient than AAGNet: for training sets below $10{,}000$ parts it consistently achieves higher PQ, reaching $\mathrm{PQ} > 0.9$ with only 250 training CAD models, whereas AAGNet requires about $5{,}000$. Finally, to demonstrate industrial applicability, we train on 270 manually labeled real-world CAD parts and show on a National Institute of Standards and Technology test sample that FeatureFox successfully performs AFR on industrial 3D-CAD geometry.

Acknowledgment

The authors thank colleagues and domain experts at Siemens AG for valuable discussions.

References

[1] Babic, B., Nesic, N., Miljkovic, Z.: A review of automated feature recognition with rule-based pattern recognition. Computers in industry **59**(4), 321–337 (2008)

[2] Colligan, A.R., Robinson, T.T., Nolan, D.C., Hua, Y., Cao, W.: Hierarchical cadnet: Learning from b-reps for machining feature recognition. Computer-Aided Design **147**, 103226 (2022)

[3] Dai, Y., Huang, X., Bai, Y., Guo, H., Gan, H., Yang, L., Shi, Y.: Brepformer: Transformer-based b-rep geometric feature recognition. In: Proceedings of the 2025 International Conference on Multimedia Retrieval. pp. 155–163 (2025)

[4] Jayaraman, P.K., Sanghi, A., Lambourne, J.G., Willis, K.D., Davies, T., Shayani, H., Morris, N.: Uv-net: Learning from boundary representations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11703–11712 (2021)

[5] Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P.: Panoptic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9404–9413 (2019)

[6] Lei, R., Wu, H., Peng, Y.: Mfpointnet: A point cloud-based neural network using selective downsampling layer for machining feature recognition. Machines **10**(12), 1165 (2022)

[7] Li, Y., Li, E., Lenover, M., Mann, S., Bedi, S.: Edge adjacency graph and neural network architecture for machining feature recognition. The International Journal of Advanced Manufacturing Technology **136**(2), 897–908 (2025)

[8] Lipman, R.R.: Nist cad models and step files with pmi (2015). https://doi.org/10.18434/mds2-1452, nIST Public Data Repository. Version: 1.0.X. Accessed: 2026-01-31

[9] Rahmani, K., Arezoo, B.: A hybrid hint-based and graph-based framework for recognition of interacting milling features. Computers in Industry **58**(4), 304–312 (2007)

[10] Shi, P., Qi, Q., Qin, Y., Scott, P.J., Jiang, X.: Intersecting machining feature localization and recognition via single shot multibox detector. IEEE Transactions on Industrial Informatics **17**(5), 3292–3302 (2020)

[11] Shi, P., Qi, Q., Qin, Y., Scott, P.J., Jiang, X.: A novel learning-based feature recognition method using multiple sectional view representation. Journal of Intelligent Manufacturing **31**(5), 1291–1309 (2020)

[12] Shi, Y., Zhang, Y., Harik, R.: Manufacturing feature recognition with a 2d convolutional neural network. CIRP Journal of Manufacturing Science and Technology **30**, 36–57 (2020)

[13] Shi, Y., Zhang, Y., Xia, K., et al.: A critical review of feature recognition techniques. Computer-Aided Design & Applications **17**(5) (2020)

[14] Subrahmanyam, S., Wozny, M.: An overview of automatic feature recognition techniques for computer-aided process planning. Computers in industry **26**(1), 1–21 (1995)

[15] Verma, A.K., Rajotia, S.: A review of machining feature recognition methodologies. International Journal of Computer Integrated Manufacturing **23**(4), 353–368 (2010)

[16] Wu, H., Lei, R., Peng, Y., Gao, L.: Aagnet: A graph neural network towards multi-task machining feature recognition. Robotics and Computer-Integrated Manufacturing **86**, 102661 (2024)

[17] Xia, M., Zhao, X., Hu, X.: Machining feature and topological relationship recognition based on a multi-task graph neural network. Advanced Engineering Informatics **62**, 102721 (2024)

[18] Zhang, H., Zhang, S., Zhang, Y., Liang, J., Wang, Z.: Machining feature recognition based on a novel multi-task deep learning network. Robotics and Computer-Integrated Manufacturing **77**, 102369 (2022)

[19] Zhang, Z., Jaiswal, P., Rai, R.: Featurenet: Machining feature recognition based on 3d convolution neural network. Computer-Aided Design **101**, 12–22 (2018)