EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

# Rapid iterative design of reconfigurable smart products via Human–AI collaboration and digital twin feedback

Zhifang Shi[1], Xue Bai[1], Jiahui Zhou[2,*]

[1]Sichuan Technology and Business University, Chengdu, 611745, China
[2]Tianjin University of Technology and Education, Tianjin, 300222, China

## Abstract

INTRODUCTION: Parametric CAD models for reconfigurable smart products require repeated modification and virtual verification. Existing CAD generation, human–AI co-design, and simulation-assisted validation methods often separate feedback analysis, constraint checking, and CAD editing, limiting the direct use of feedback for executable refinement.
OBJECTIVES: This study develops a closed-loop CAD refinement framework that converts design feedback and digital twin–based simulation or constraint feedback into coordinated edit actions while reducing constraint violations, ineffective revisions, and oscillatory updates.
METHODS: A conflict-aware dual-feedback framework is proposed. Design feedback is translated into target parameters, local regions, modification directions, and command-level edit conditions. Simulation and constraint results are written back as violation-aware correction cues. When feedback sources conflict, active constraints, historical modification states, and edit magnitudes are jointly considered to generate controlled parameter and operation-sequence updates. The framework is evaluated on feedback-driven benchmarks constructed from the public DeepCAD and SketchGraphs datasets.
RESULTS: On the constructed DeepCAD benchmark, the proposed method achieved a constraint satisfaction rate of 90.6% and an average iteration count of 3.43. On the constructed SketchGraphs benchmark, it obtained 89.4% and 3.61, respectively. It outperformed the strongest comparison methods in constraint satisfaction and iterative efficiency while maintaining competitive CAD validity and edit-action accuracy. Statistical and ablation analyses confirmed the contributions of feedback translation, correction write-back, and conflict-aware coordination.
CONCLUSION: The results demonstrate the effectiveness of the proposed closed-loop refinement framework under the constructed feedback-driven benchmark settings, particularly in improving constraint satisfaction, reducing refinement rounds, and maintaining stable optimization under conflicting feedback.

**Keywords:** reconfigurable smart products, CAD refinement, human–AI collaboration, constraint feedback, digital twin validation

Received on 27 July 2026, accepted on 17 August 2026, published on 22 September 2026



doi: 10.4108/eetsis.14186

## 1. Introduction

This paper investigates rapid iterative parametric CAD design for reconfigurable smart products [1,2]. These products have adjustable parameters, configurable functions, or modular structures, with emphasis on parametric and structural reconfiguration in CAD design. For smart-device housings, connectors, brackets, and modular components, design commonly requires repeated updates of parameters, local structures, and operation sequences based on design feedback and virtual validation results [3-5]. Here, digital twin feedback is represented during CAD refinement by virtual validation signals from CAD-state evaluation and simulation/constraint checking [5].

*Corresponding author. Email: 2023080001@tute.edu.cn

Existing studies on parametric CAD generation, human–AI co-design, and digital-twin- or simulation-based validation provide important foundations for CAD generation, feedback acquisition, constraint-driven optimization, and candidate-solution evaluation [5-7]. However, these capabilities are commonly addressed as separate stages or are used primarily for single-pass generation, interaction, validation, or solution selection [8,9]. The central challenge in multi-round refinement is therefore not simply to combine heterogeneous feedback sources, but to transform them into edit-oriented representations that can directly support executable CAD modification [10]. In particular, design feedback must be grounded to editable CAD objects and modification directions, while simulation or constraint feedback must be converted from an evaluation signal into a correction cue for the next refinement round. Conflicts between preference-driven and constraint-driven modifications further require coordination at the CAD edit-action level rather than simple feedback fusion.

To address this gap, this paper proposes a conflict-aware dual-feedback CAD refinement method for modifying existing designs over multiple rounds. The method converts design and simulation/constraint feedback into complementary edit-oriented conditions and uses them to determine the next CAD modification. Simulation and constraint results are written back into the refinement loop rather than being used only for ranking, filtering, or post-generation validation. When the two feedback sources imply inconsistent modifications, conflict-aware coordination prioritizes active engineering constraints while accounting for the historical modification state. A history-aware damping mechanism further suppresses repeated directional reversals and excessive edit magnitudes, thereby reducing oscillatory refinement.

The contributions are fourfold. First, feedback-conditioned CAD refinement is formulated as an iterative edit-action prediction problem driven jointly by the current CAD state, heterogeneous feedback, and modification history. Second, an edit-oriented feedback representation is introduced to connect design intent with executable CAD modification. Third, simulation and constraint validation are incorporated as next-round correction cues, forming a closed feedback-to-edit loop rather than a post-hoc verification process. Fourth, conflict-aware coordination with history-aware damping is introduced to manage inconsistent feedback directions and reduce repeated or oscillatory updates. Using feedback-driven benchmarks constructed from DeepCAD and SketchGraphs, the proposed method is evaluated through main experiments, statistical analysis, failure analysis, and ablation studies.

## 2. Related Work

This section reviews research on rapid iterative CAD design for reconfigurable smart products from four perspectives: parametric CAD generation and editable modeling, human–AI co-design and feedback modeling, digital twin and simulation-based validation, and the remaining research gap. Existing studies provide foundations for CAD generation, feedback interaction, interactive optimization, preference modeling, and virtual validation, but do not fully address how heterogeneous feedback can be transformed into coordinated next-round CAD edit actions during multi-round refinement.

### 2.1. Parametric CAD Generation and Editable CAD Modeling

Parametric CAD generation and editable modeling provide technical support for engineering design automation [11,12]. Existing methods generally fall into three categories. The first represents CAD modeling as operation sequences, such as sketching and extrusion, to support command-level generation [13]. The second maps text, sketches, or other conditional inputs to parametric CAD structures [14,15]. The third improves geometric representation and CAD reasoning through B-rep modeling and advanced CAD generation or assistance [16,17]. DeepCAD, Text2CAD, VQ-CAD, SolidGen, and BrepGen are representative methods. CAD editing, repair, interactive modification, and simulation-in-the-loop refinement are also relevant because they modify existing CAD states rather than generate models only once.

Recent LLM- and VLM-based methods further extend CAD generation toward natural-language interaction and design-intent interpretation. FlexCAD uses fine-tuned large language models for controllable CAD generation, CAD-assistant employs tool-augmented vision-language models for general CAD tasks, and CADialogue supports multimodal conversational interaction with parametric CAD operations [12,16,20]. These methods strengthen natural-language-to-CAD generation, command execution, and interactive design assistance. However, their primary objective remains generating or executing CAD operations from user instructions rather than coordinating design preferences with simulation or engineering-constraint feedback across successive refinement rounds.

Most existing CAD generation methods also emphasize single-pass generation and evaluate syntactic validity, geometric constructibility, or sequence plausibility [18,19]. For reconfigurable smart products, valid generation alone is insufficient because parameters and operation sequences must be repeatedly updated according to design feedback, constraint-validation results, and modification history. Existing methods are therefore not primarily designed for joint decision-making among the current CAD state, heterogeneous feedback, and historical modifications. This study shifts the focus from one-shot generation toward feedback-conditioned CAD edit-action refinement.

### 2.2. Human–AI Co-design and Design Feedback Modeling

Human–AI co-design research examines how AI supports designers during ideation, comparison, and modification [20,21]. COFI analyzes co-creative interaction through factors such as turn-taking, communication, and contribution.

Research on nonlinear collaboration further shows that design involves exploration, reflection, comparison, and recombination rather than a simple input–generation process. Systems including AIdeation, StepIdeator, and Sketch2Prototype demonstrate the value of generative AI for conceptual ideation, sketch exploration, and design expansion [21–23].

Interactive optimization and preference learning provide a related mechanism for incorporating designer judgments into iterative design. Koyama and Goto [24] use Bayesian optimization as an asynchronous design assistant that infers preferences from user interaction and generates parameter suggestions, while Nandy and Goucher-Lambert [25] investigate interactive preference learning for adapting designs to subjective semantic attributes. These approaches demonstrate that human evaluations can progressively alter the search process rather than remain fixed input conditions. However, their main purpose is to optimize candidate selection, parameter search, or learned preference functions. They do not explicitly map a preference to an affected CAD object and edit direction while simultaneously converting engineering violations into command-level correction actions.

Conventional human–AI co-design studies likewise mainly address conceptual design, interaction experience, or early-stage exploration. Feedback is commonly expressed as preferences, evaluations, selections, or conversational instructions and is used to guide generation or compare alternatives. Its explicit conversion into CAD parameter changes, local structural edits, or command-level conditions remains limited. Although CADialogue connects natural-language interaction with CAD commands [20], the joint grounding of feedback semantics to affected parameters, local regions, edit directions, and constraint-aware next-round modifications remains insufficient. This study therefore focuses on converting design feedback into edit-oriented conditions that can directly participate in multi-round CAD refinement.

## 2.3. Digital Twin and Simulation-Based Design Validation

Digital twins and simulation-based methods support virtual verification and constraint evaluation in product design [26,27]. Digital-twin-driven design frameworks integrate virtual models, product states, and validation information into product development, while recent studies extend digital twins from manufacturing monitoring and maintenance toward engineering design [28,29]. Human-in-the-loop digital twin research further emphasizes coordination among human, cyber, and physical spaces in complex design and decision processes [30,31].

Simulation-based validation and digital-twin-driven virtual evaluation further connect candidate modification with repeated engineering evaluation [32,33]. Such approaches can use feasibility results, objective values, constraint violations, gradients, or candidate rankings to steer subsequent optimization. However, these signals commonly support evaluation, filtering, ranking, or numerical parameter optimization and do not necessarily identify which CAD parameter, local structure, or discrete modeling operation should be modified next. Consequently, a constraint violation may influence the optimization objective without being represented as an explicit command-oriented correction cue for the next CAD refinement round.

This study instead treats simulation and constraint feedback as an internal refinement signal. Constraint violations, validation deviations, and feasibility results are linked to affected CAD objects and correction directions so that they can directly condition the next-round edit action rather than merely accept, reject, rank, or numerically optimize a completed candidate.

## 2.4. Summary and Research Gap

Existing research establishes important foundations for CAD generation, human-guided parametric editing, LLM/VLM-based interaction, interactive optimization, preference learning, constraint-driven optimization, and digital-twin validation. However, these research streams address different parts of the iterative design process. CAD generation and LLM/VLM-based interaction mainly support one-shot generation, design-intent interpretation, feedback acquisition, or command execution [11–20]. Human–AI co-design supports ideation and interactive modification [21–23], while interactive optimization and preference learning mainly use human evaluations to guide candidate search or learn subjective objectives [24,25]. Simulation-in-the-loop, constraint-driven optimization, and digital-twin approaches primarily support feasibility checking, candidate ranking, numerical optimization, and virtual validation [26–33].

These differences are important because multi-feedback fusion alone does not define how heterogeneous feedback should produce an executable next-round CAD modification. In the proposed framework, design feedback is first grounded to an editable CAD parameter or local region together with a modification direction and operation condition. A simulation or constraint violation is then converted into an edit-oriented correction cue rather than retained only as an evaluation, filtering, ranking, or optimization signal. When preference-driven and constraint-driven feedback imply incompatible modifications, they are coordinated at the CAD edit-action level, while modification history is used to suppress repeated directional reversals. The proposed method therefore differs from generic multi-feedback combination by constructing explicit intermediate representations linking feedback to executable CAD edits.

Accordingly, this study formulates rapid iterative design as a dual-feedback-driven parametric CAD refinement task. It addresses three questions: how to translate design feedback into executable CAD edit conditions, how to write simulation and constraint feedback back as next-round correction cues, and how to coordinate edit actions when design preferences conflict with engineering constraints. Its distinction lies in connecting feedback acquisition, violation correction, and conflict resolution within a closed feedback-to-edit loop

rather than treating generation, preference optimization, validation, and feedback fusion as separate stages.

## 3. Methodology

### 3.1. Problem Formulation

This study addresses iterative modification of parametric CAD models for reconfigurable smart products, including smart-device housings, connectors, brackets, and modular components. The task is to generate the next-round CAD edit action from the current CAD state, design feedback, simulation/constraint feedback, and historical modification state. The objective is to progressively satisfy geometric validity, engineering constraints, and feedback requirements within limited iterations. The study focuses on feedback-driven refinement of existing CAD designs rather than generation from scratch. The formulation is intended to accommodate design feedback and simulation or constraint feedback generated in practical CAD refinement environments. In the experimental implementation of this study, however, these feedback signals are reconstructed from parameters, operation records, geometric relations, and constraint information available in the public DeepCAD and SketchGraphs datasets, rather than collected from complete industrial revision sessions.

Let the CAD state of the i-th sample at the t-th iteration be defined as:

$$C_i^t = (p_i^t, s_i^t) \tag{1}$$

where $p_i^t$ denotes the parameter vector and $s_i^t$ denotes the CAD operation sequence. The design feedback, simulation/constraint feedback, and historical state are denoted as $F_i^t$, $V_i^t$, and $H_i^t$, respectively. The model input is:

$$X_i^t = \{C_i^t, F_i^t, V_i^t, H_i^t\} \tag{2}$$

The output CAD edit action is:

$$A_i^t = (\Delta p_i^t, \Delta s_i^t) \tag{3}$$

where $\Delta p_i^t$ represents parameter modification and $\Delta s_i^t$ represents operation-sequence modification. After executing $A_i^t$, the CAD state is updated as:

$$C_i^{t+1} = U(C_i^t, A_i^t) \tag{4}$$

where $U(\cdot)$ denotes the CAD update operation. The updated CAD design should satisfy the engineering constraints:

$$g_k(C_i^{t+1}) \le 0, k = 1, \dots, K \tag{5}$$

where $g_k(\cdot)$ is the k-th constraint function and K is the number of constraints. The stopping condition is reached when the CAD model is constructible and the main constraints are satisfied, or when the maximum iteration number is reached.

Uncertainty arises from incomplete design feedback, indirect simulation/constraint signals, and conflicts between preference- and constraint-driven directions. The optimization objective is therefore to predict valid edits, reduce constraint violations, satisfy feedback requirements, and minimize refinement rounds.

### 3.2. Overall Framework

As shown in Figure 1, the proposed framework organizes iterative CAD refinement into three successive stages: feedback grounding, correction write-back, and conflict-aware edit coordination. These stages transform heterogeneous feedback into edit-oriented information and then generate the next-round CAD edit action using the current CAD state and historical modification state.

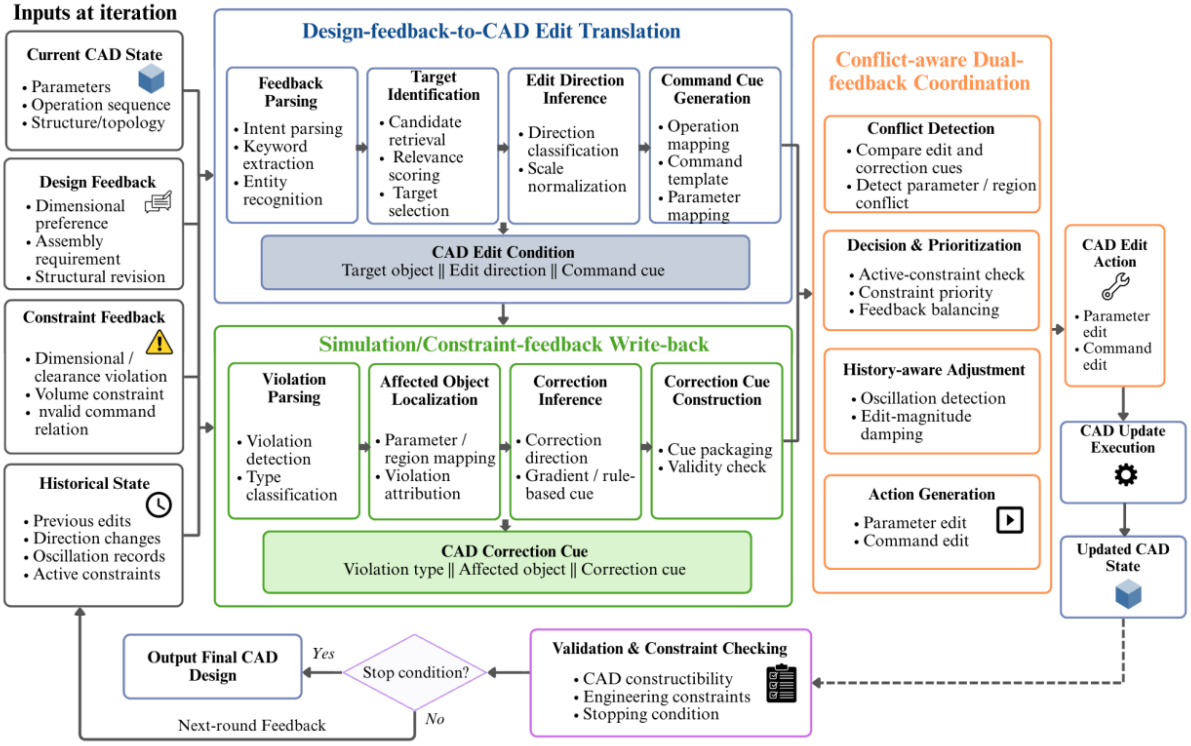


**Figure 1.** Overall framework of conflict-aware dual-feedback CAD refinement

In the feedback-grounding stage, design feedback $F_i^t$ is transformed into CAD edit conditions $E_i^t$ associated with editable parameters, local regions, modification directions, and operation cues. In the correction write-back stage, simulation/constraint feedback $V_i^t$ is converted into correction cues $Q_i^t$ that identify constraint violations, affected CAD objects, and correction directions. In the conflict-aware edit-coordination stage, $E_i^t$ and $Q_i^t$ are coordinated with the current CAD state $C_i^t$ and historical state $H_i^t$ to produce the final edit action $A_i^t$. Constraint-related corrections take priority when feedback directions conflict, while repeated directional reversals are moderated using the historical modification state.

The overall data flow is:

$$(C_i^t, F_i^t, V_i^t, H_i^t) \rightarrow (E_i^t, Q_i^t) \rightarrow A_i^t \rightarrow C_i^{t+1} \quad (6)$$

where $E_i^t$, $Q_i^t$, and $A_i^t$ denote the CAD edit condition, correction cue, and final edit action, respectively. The process continues until the stopping condition or maximum iteration number is reached. Detailed implementations of feedback grounding, correction write-back, and conflict-aware coordination are presented in Sections 3.3–3.5.

## 3.3. Design-Feedback-to-CAD Edit Translation

The design-feedback-to-CAD edit translation module converts design feedback into executable CAD edit conditions. As shown in Figure 2, feedback is grounded to editable CAD parameters or local regions and mapped to modification directions and command-level operations. The module uses a deterministic preprocessing and matching pipeline rather than an additional pretrained language model.

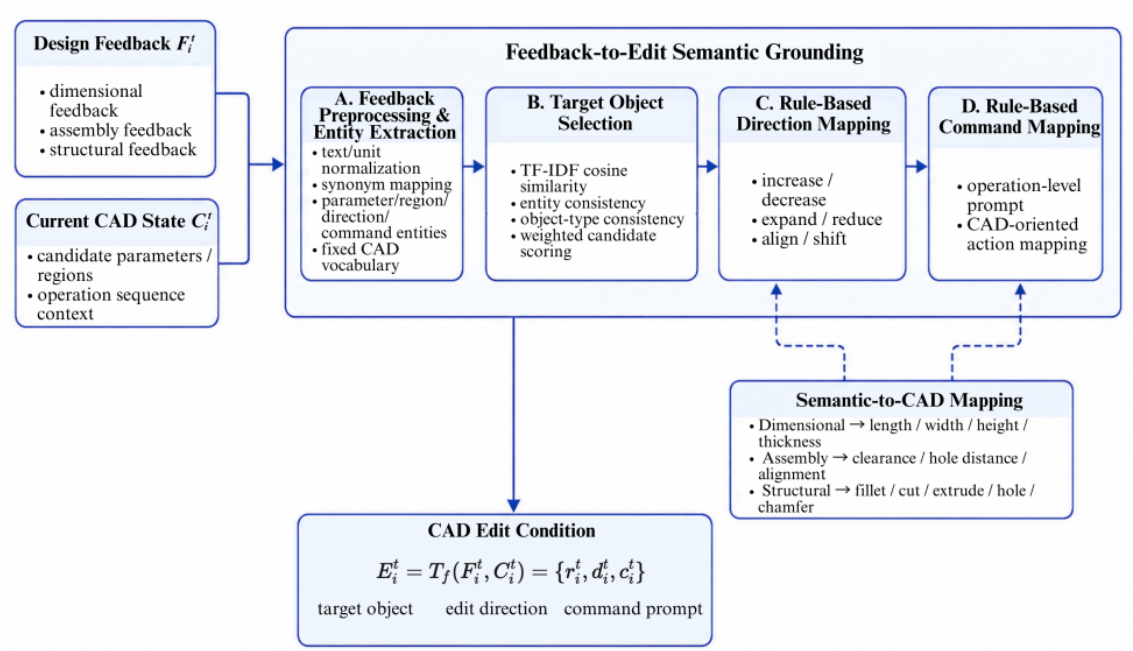


**Figure 2.** Design-feedback-to-CAD edit translation module

In the current implementation, design feedback is represented as textual feedback rather than raw image feedback. Given design feedback $F_i^t$ and the current CAD state $C_i^t$, text is first normalized by lowercasing, standardizing numerical values and engineering units, removing non-informative tokens, and mapping synonymous expressions to a fixed CAD vocabulary. Feedback entities are then extracted using parameter, region, direction, and command dictionaries derived from the CAD schemas of DeepCAD and SketchGraphs. Each editable candidate is represented as $o_{i,j}^t = (l_{i,j}, r_{i,j}, c_{i,j})$, where $l_{i,j}$, $r_{i,j}$, and $c_{i,j}$ denote the parameter label, local-region label, and associated CAD command, respectively.

The target editable object is selected by combining semantic similarity, entity consistency, and object-type consistency:

$$\arg\max_{o_{i,j}^t \in O_i^t} [\lambda_s \cos(z_F, z_{i,j}) + \lambda_e I_{\text{ent}} + \lambda_t I_{\text{type}}], \quad (7)$$

where $z_F$ and $z_{i,j}$ are TF-IDF vectors of the normalized feedback and candidate labels. $I_{\text{ent}}$ indicates whether extracted entities match the candidate parameter or region, and $I_{\text{type}}$ indicates feedback-object type consistency. The weights are set to $\lambda_s = 0.60$, $\lambda_e = 0.25$, and $\lambda_t = 0.15$. TF-IDF statistics are fitted only on the training split and remain fixed during validation and testing.

The CAD edit condition is defined as

$$E_i^t = (o_i^{t*}, d_i^t, c_i^t), \quad (8)$$

where $d_i^t$ is the modification direction and $c_i^t$ is the command-level edit condition. They are obtained by

$$(d_i^t, c_i^t) = R_F(e_i^t, \tau_i^t, \rho_i^t), \quad (9)$$

where $e_i^t$ denotes extracted entities, $\tau_i^t$ denotes feedback type, $\rho_i^t$ denotes an explicit magnitude or target value when available, and $R_F$ is a fixed feedback-to-edit rule set. Dimensional feedback is mapped to parameter increase or decrease, assembly feedback to clearance, distance, oralignment adjustment, and structural feedback to operations such as fillet, chamfer, cut, hole, or extrusion. If no numerical target is provided, the qualitative direction is retained and the final edit magnitude is determined by the downstream action predictor.

No pretrained language model is used in this module. The dictionaries and rule mappings are fixed and receive no gradient updates. Only the TF-IDF representation is fitted on the training split. Target objects, directions, and command labels from the constructed benchmark supervise downstream edit-action learning, while the vocabulary, IDF statistics, and rule mappings remain frozen during inference.

## 3.4. Simulation/Constraint-Feedback-to-CAD Correction Write-Back

The simulation/constraint-feedback-to-CAD correction write-back module converts validation results into executable correction cues. As shown in Figure 3, each detected violation is linked to an affected CAD object and a numerical or discrete correction, allowing validation information to guide the next refinement round rather than serving only as a ranking or filtering signal.

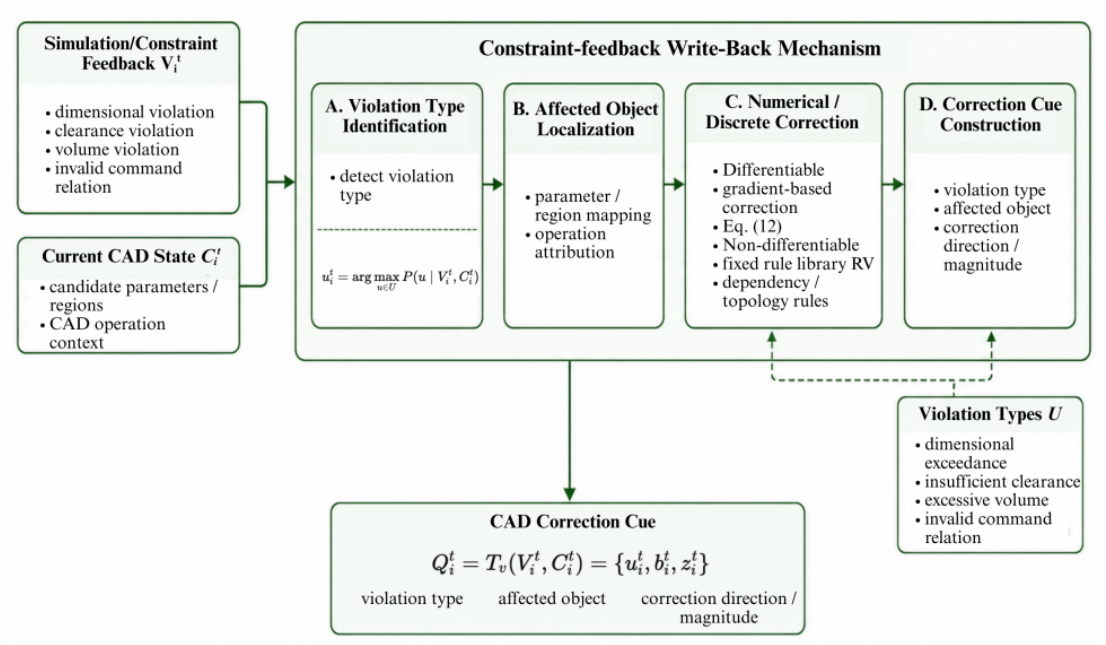


**Figure 3.** Simulation/Constraint-Feedback-to-CAD Correction Write-Back Module

Given simulation/constraint feedback $V_i^t$ and CAD state $C_i^t$, the violation type is identified as

$$v_i^t = D_V(V_i^t, C_i^t), \quad v_i^t \in \mathcal{V}, \tag{10}$$

where $\mathcal{V}$ includes dimensional, clearance, alignment, volume, topology, and command-dependency violations. The correction cue is

$$Q_i^t = \left(v_i^t, o_{v,i}^t, \delta_{v,i}^t\right), \tag{11}$$

where $o_{v,i}^t$ is the affected parameter, local object, or command token, and $\delta_{v,i}^t$ specifies the correction direction or discrete edit. For differentiable numerical constraints,

$$-\eta_v \frac{\nabla g_k(C_i^t)}{|\nabla g_k(C_i^t)|_2 + \epsilon}, \tag{12}$$

where $\eta_v$ is the correction step coefficient and $\epsilon$ prevents numerical instability.

For non-differentiable CAD relations, correction is determined by a fixed rule library $\mathcal{R}_V = r_m$, where each rule is represented as $r_m = (T_m, O_m, P_m, A_m, \pi_m)$. Here, $T_m$ denotes the violation trigger, $O_m$ the affected CAD object, $P_m$ the rule precondition, $A_m$ the replacement, reordering, or parameter-edit action, and $\pi_m$ the priority when multiple violations are active. Constraint-safety rules receive higher priority than preference-only modifications. Table 1 summarizes representative correction rules for the main violation types considered in the current benchmark.

Table 1. Representative constraint-feedback correction rules

| Violation | Trigger / condition | Affected object | Rule | Correction cue |
|---|---|---|---|---|
| Dimensional lower-bound violation | Parameter below permitted minimum | Length, width, thickness, or radius | Bound-restoration rule | Increase parameter |
| Insufficient clearance | Clearance below required threshold | Slot width or clearance region | Clearance rule | Increase clearance |
| Alignment/distance violation | Relative position exceeds tolerance | Hole center or assembly parameter | Geometric-relation rule | Shift toward target relation |
| Invalid chamfer/fillet dependency | Parent feature unavailable | Command token and sequence position | Command-dependency rule | Reorder command |
| Invalid extrusion dependency | Valid profile or parent operation missing | Extrusion command token | Operation-precondition rule | Reorder or suppress operation |

The rule library covers numerical parameter bounds, assembly/geometric relations, topology conditions, and

discrete command dependencies represented in the benchmark. Differentiable constraints are handled by Eq. (12), whereas topology and sequence violations use the rule library. Unsupported violations are marked unresolved rather than assigned an arbitrary correction.

A representative non-differentiable case occurs when a chamfer command is generated before the extrusion that creates its target edge. Because command order is discrete, a gradient cannot directly determine how the operation tokens should be reordered. The dependency rule identifies the missing parent feature and moves the chamfer after the required extrusion while retaining its geometric parameters. The corrected sequence is then revalidated before the next refinement round.

Thus, the write-back mechanism records not only whether a constraint is violated, but also the affected CAD object and the required numerical or discrete correction for the next edit action.

## 3.5. Conflict-Aware Dual-Feedback Coordination Mechanism

The conflict-aware dual-feedback coordination mechanism integrates the edit condition and correction cue to generate the final CAD edit action. Conflicts may include directional, interval, partial, and multi-constraint conflicts. The current implementation explicitly resolves parameter-level directional and range-overlap conflicts, while more complex interactions are handled through active-constraint priority and iterative revalidation.

Given $E_i^t$, $Q_i^t$, $C_i^t$, and $H_i^t$, a directional conflict is detected when the design- and constraint-driven updates for the same object have opposite signs:

$$M_i^t = \{m \mid e_{i,m}^t q_{i,m}^t < 0\}. \tag{13}$$

Here, $m$ denotes a parameter or local modification object, while $e_{i,m}^t$ and $q_{i,m}^t$ denote the corresponding design- and constraint-driven directions. An interval conflict is activated when the design-preferred target falls outside the feasible range of an active constraint. When multiple constraints affect the same object, safety-related constraints are processed first, followed by geometric or assembly constraints and preference-only edits. Partial conflicts are resolved locally without suppressing compatible edits on other parameters.

To suppress repeated directional reversals, the history-aware damping factor is

$$\gamma_{i,m}^t = \frac{1}{1+\kappa o_{i,m}^t}, \tag{14}$$

where $o_{i,m}^t$ is the number of recorded direction changes and $\kappa$ controls damping strength. The parameter update is

$$\Delta p_{i,m}^t = \gamma_{i,m}^t \begin{cases} q_{i,m}^t, & m \in M_i^t \text{ and an active constraint exists,} \\ \alpha e_{i,m}^t + (1-\alpha) q_{i,m}^t, & m \notin M_i^t \text{ and an active constraint exists,} \\ \beta e_{i,m}^t, & \text{no active constraint exists.} \end{cases} \tag{15}$$

Here, $\alpha$ controls coordination between compatible feedback sources and $\beta$ controls the design-feedback edit magnitude. The final action is $A_i^t = (\Delta p_i^t, \Delta s_i^t)$.

The direction-change count is adopted as a low-complexity damping signal because it requires no additional controller training and introduces negligible inference overhead. Unlike momentum or adaptive step-size methods, it directly reflects oscillatory edit history. The ablation results in Section 4.7 show that removing this mechanism increases the average iteration count on both datasets. The current mechanism is nevertheless limited in modeling nonlinear dependencies among multiple CAD objects; multi-criteria decision making, learning-to-rank, or adaptive controllers could provide more expressive conflict resolution in future work.

## 3.6. Objective Function and Training Strategy

The framework separates fixed feedback-processing mechanisms from trainable edit-action prediction. The TF-IDF representation, CAD dictionaries, feedback-to-edit rule set $R_F$, correction rule library $R_V$, constraint priorities, and history-aware damping remain fixed. Trainable components include the CAD-state representation layers and the parameter- and operation-sequence prediction branches.

Training supervision is constructed from DeepCAD and SketchGraphs samples and includes target parameter modifications and operation-sequence edits. Target object, direction, and command labels are used to construct and evaluate the feedback-grounding conditions, while gradient-based optimization is applied only to the trainable prediction components.

The total loss is

$$\lambda_a \mathcal{L}\text{act} + \lambda_c \mathcal{L}\text{con} \tag{16}$$

where $\mathcal{L}$act supervises parameter and sequence edits and $\mathcal{L}$con penalizes engineering-constraint violations. The coefficients are selected on the validation set according to CVR, CSR, and AIC.

The action-prediction loss is

$$\left|\Delta p_i^t - \Delta p_i^{t,}\right| 1 + \mathcal{L}\text{seq}\left(\Delta s_i^t, \Delta s_i^{t,}\right), \tag{17}$$

where $\Delta p_i^{t,}$ and $\Delta s_i^{t,}$ are the target parameter and operation-sequence edits. The constraint loss is

$$\sum_{k=1}^{K} \max\left(0, g_k(C_i^{t+1})\right) \tag{18}$$

Training proceeds in two stages. First, the CAD representation and edit-action branches are pretrained on single-step edit pairs. They are then jointly fine-tuned using

Eq. (16), with the constraint penalty evaluated after each predicted CAD update. The TF-IDF statistics, dictionaries, rule mappings, priority rules, and damping mechanism remain frozen throughout training. Validation data are used for hyperparameter selection and early stopping, and test data are excluded from fitting and model selection.

## 3.7. Stability Considerations and Bounded Update Analysis

Uncertainty arises from ambiguous design feedback, indirect constraint signals, and conflicting modification directions. Stability is supported by constraint penalization, active-constraint priority, and history-aware damping.

When CAD parameters and feedback-driven updates are bounded, the per-step edit magnitude satisfies

$$\left|\Delta p_{i,m}^{t}\right| \le \gamma_{i,m}^{t} B_{\Delta}, \quad 0 < \gamma_{i,m}^{t} \le 1, \tag{19}$$

where $B_{\Delta}$ is the undamped upper bound. Equation (19) establishes bounded edit magnitude rather than asymptotic convergence. Since $T_{\max}$ is fixed, inference terminates within a finite number of refinement steps.

Under bounded stochastic gradients and fixed batch size (B),

$$\mathrm{Var}(\nabla \mathcal{L}_{B}) \le \frac{\sigma^{2}}{B}, \tag{20}$$

where $\sigma^{2}$ is the upper bound of single-sample gradient variance. Equation (20) characterizes training-time variance under the stated assumptions but does not constitute a global convergence guarantee. The analysis is therefore limited to bounded updates, finite-step termination, and controlled training variance, with empirical stability further evaluated in Section 4.

## 3.8. Algorithm Procedure

Algorithm 1 summarizes the training and iterative inference procedure. TF-IDF statistics are fitted on the training split, while the feedback dictionaries and rule libraries remain fixed. Only the CAD representation and edit-action prediction components are optimized.

Algorithm 1. Conflict-aware dual-feedback CAD refinement

Input: Initial CAD state $C_i^0$, design feedback $F_i^t$, simulation/constraint feedback $V_i^t$, maximum iterations $T_{\max}$
Output: Final CAD state $C_i^T$
1: Fit training-split TF-IDF statistics and freeze $R_F$ and $R_V$.
2: Pretrain the CAD representation and edit-action prediction branches.
3: Jointly fine-tune trainable components using Eq. (16).
4: Initialize $t = 0$ and $H_i^0 = \emptyset$.
5: while $t < T_{\max}$ do
6: Generate edit condition $E_i^t$ using Section 3.3.
7: Identify violation $v_i^t$ and affected object $o_{v,i}^t$.
8: if differentiable, compute correction using Eq. (12); else apply $R_V$.
9: Form $Q_i^t$ and detect directional, interval, or priority conflicts.
10: Apply history-aware damping and generate $A_i^t$ using Eqs. (14)–(15).
11: Update $C_i^{t+1} = U(C_i^t, A_i^t)$ and revalidate validity and constraints.
12: Update $H_i^{t+1}$.
13: if $C_i^{t+1}$ is valid and main constraints are satisfied, break.
14: Set $t = t + 1$.
15: end while
16: return $C_i^T$.

The algorithm shows that the proposed method is not one-shot CAD generation or post-filtering. It forms a closed-loop refinement process that repeatedly transforms design feedback, constraint feedback, and historical states into executable CAD edit actions.

# 4. Experiments and Results

## 4.1. Experimental Setup

### 4.1.1 Datasets

DeepCAD and SketchGraphs are used to construct feedback-driven CAD refinement benchmarks. DeepCAD provides parametric construction sequences and numerical representations, whereas SketchGraphs provides geometric entities and constraint relations. Because complete designer feedback and industrial revision logs are unavailable, feedback-to-edit instances are reconstructed from CAD states, editable parameters, operation sequences, and constraints.

Source CAD models or sketches are assigned to only one train/validation/test split to prevent leakage. Invalid, duplicate, incomplete, or non-executable samples are removed. DeepCAD operations are parsed into editable parameters and commands, while SketchGraphs entities and relations are represented as structured constraint states.

For each retained sample, one primary editable parameter, region, or valid command is selected while its active constraints are retained. Design feedback covers dimensional adjustment, shape preference, assembly-clearance requirement, and structural/command preference. Continuous parameters are normalized to [0,1] and perturbed by 5–20% of their feasible span. Positive directions denote increase, outward shift, enlargement, or later adjustment, whereas negative directions denote decrease, inward shift, reduction, or earlier adjustment. Discrete commands use add, remove, replace, or reorder labels.

Constraint feedback is generated by perturbing a feasible CAD state and rechecking its constraints. Numerical violations move parameters 5–15% beyond active bounds or tolerances. Clearance and alignment violations cross their

relation thresholds, volume violations exceed an upper bound, and topology or command-dependency violations are produced by modifying valid operation sequences. Only initially valid and executable samples are used for corruption.

Ground-truth edit actions are derived from the original feasible state and the injected perturbation. Numerical targets identify the affected parameter and inverse correction direction, whereas discrete targets specify the required command replacement or reordering. For dual-feedback cases, feasibility is enforced before selecting the feasible edit closest to the design preference. Table 2 summarizes the procedure. After construction, 126,420 DeepCAD and 162,800 SketchGraphs samples are retained using a 70/15/15 split, as summarized in Table 3.

Table 2. Benchmark construction procedure and label generation

| Feedback/Violation Type | Sampling Source | Perturbation | Generated Feedback | Target Label | Validity Check |
|---|---|---|---|---|---|
| Dimensional preference | Editable numerical parameter | ±5–20% feasible span | Increase/decrease request | Parameter + direction | Parameter bounds |
| Shape/structural preference | Local feature or command | Parameter/command modification | Structural request | Object + command | CAD validity |
| Clearance requirement | Gap or clearance relation | Cross threshold | Clearance violation | Parameter + direction | Clearance check |
| Alignment/distance violation | Related entities | Cross tolerance | Relation violation | Object + shift | Geometric check |
| Bound/volume violation | Size parameter | 5–15% beyond bound | Bound/volume violation | Parameter + correction | Bound/volume check |
| Command dependency | Valid operation sequence | Remove/replace/reorder | Invalid dependency | Command correction | Sequence execution |

Table 3. Dataset and benchmark statistics

| Dataset | Records Used | Retained Samples | Train / Val / Test | Main Derived Labels |
|---|---|---|---|---|
| DeepCAD | 178,238 | 126,420 | 88,494 / 18,963 / 18,963 | Parameter, command, direction |
| SketchGraphs | 250,000 | 162,800 | 113,960 / 24,420 / 24,420 | Violation, object, correction cue |
| Combined | 428,238 | 289,220 | 202,454 / 43,383 / 43,383 | Feedback and target edit action |

To avoid circular evaluation, feedback-generation constraints create violation cues, target-generation constraints determine correction labels, and evaluation constraints are re-executed on the final state $C_i^T$. CSR therefore measures whether the refined CAD state satisfies the required constraints rather than whether the model reproduces the input constraint label. Test instances are excluded from TF-IDF fitting, rule construction, and parameter optimization.

### 4.1.2 Baselines

The proposed method is compared with ten baselines covering procedural reconstruction, sketch-based modeling, command parsing, CAD sequence prediction, verification-assisted generation, vision-language reconstruction, constraint generation, and LLM-based CAD generation. FlexCAD [12] is added as a recent LLM-based baseline. Because it targets controllable CAD generation rather than dual-feedback refinement, FlexCAD is adapted to the same executable CAD-edit evaluation protocol used for the other baselines. Table 4 summarizes the compared methods, their methodological categories, and their roles in the experimental comparison.

Table 4. Compared methods

| Method | Type | Reason |
|---|---|---|
| Fusion 360 Gallery [34] | Procedural CAD reconstruction | Human design sequence |
| Sketch2CAD [35] | Sketch-based CAD modeling | Sketch-conditioned editing |
| Free2CAD [36] | Freehand-to-CAD commands | Command parsing |
| SketchGen [37] | Constraint sketch generation | Constraint reference |
| TransCAD [38] | CAD sequence inference | Transformer prediction |
| CADCodeVerify [8] | Verification-based CAD generation | Verification-assisted generation |
| CAD2Program [39] | Vision-language CAD reconstruction | 2D-to-3D reconstruction |
| Drawing2CAD [40] | Drawing-to-CAD sequence | Vector-to-command generation |
| Aligning Constraint Generation with Design Intent [19] | Constraint generation | Design-intent modeling |
| FlexCAD [12] | LLM-based CAD generation | Recent LLM baseline |
| Proposed method | Dual-feedback refinement | Feedback-to-edit refinement |

#### 4.1.3 Implementation Details

Experiments are implemented in Python with PyTorch on an Intel i7-12700K CPU, 32 GB RAM, and an NVIDIA RTX 3060 GPU. CAD parameters are normalized to $[0,1]$, and operation sequences are tokenized by command type, parameter field, and order.

The grounding weights are $\lambda_s = 0.60$, $\lambda_e = 0.25$, and $\lambda_t = 0.15$. TF-IDF statistics, CAD dictionaries, $R_F$, $R_V$, constraint-priority rules, and history-aware damping remain fixed after training-split construction. Only the CAD-state representation and parameter/operation prediction branches are trainable.

The trainable components are first pretrained on single-step edit pairs and then jointly fine-tuned using Eq. (16). The maximum iteration number is $T_{\max} = 5$, with batch size 16, AdamW, learning rate $1 \times 10^{-4}$, and at most 100 epochs. Early stopping is applied after 15 unchanged validation epochs. $\lambda_a$ and $\lambda_c$ are selected on the validation set. Experiments are repeated five times using seeds 2022–2026, with results reported as mean ± standard deviation.

Training time, peak GPU memory, and average per-iteration inference time were measured on the same RTX 3060 platform, with offline preprocessing and benchmark construction excluded from inference timing. The resulting training time, peak GPU memory, and average per-iteration inference time were 6.4 h, 7.2 GB, and 86 ms, respectively.

#### 4.1.4 Evaluation Metrics

Four metrics are used. CAD Validity Rate (CVR) measures whether the final CAD model is executable and constructible. Constraint Satisfaction Rate (CSR) measures whether the final state satisfies dimensional, assembly, geometric, volume, topology, and command constraints. CSR is obtained by rechecking $C_i^T$, rather than comparing the prediction with the input constraint-feedback label.

Edit Action Accuracy (EAA) measures agreement between the predicted and target parameter or operation-sequence edit. Average Iteration Count (AIC) measures the refinement rounds required to reach the stopping condition; unsuccessful samples are assigned $T_{\max}$.

### 4.2. Main Experimental Results

Tables 5 and 6 report the results on DeepCAD and SketchGraphs, respectively. The proposed method is compared with ten baselines using CVR, CSR, EAA, and AIC. Overall, the proposed framework shows its main advantages in constraint satisfaction and iterative efficiency while maintaining competitive CAD validity and edit-action accuracy.

Table 5. Main experimental results on DeepCAD

| Method | CVR ↑ | CSR ↑ | EAA ↑ | AIC ↓ |
|---|---|---|---|---|
| Fusion 360 Gallery | 90.8 ± 0.7 | 84.6 ± 0.9 | 78.7 ± 0.8 | 4.74 ± 0.22 |
| Sketch2CAD | 88.5 ± 0.8 | 82.8 ± 1.0 | 77.2 ± 0.9 | 4.93 ± 0.26 |
| Free2CAD | 87.6 ± 1.0 | 81.5 ± 1.1 | 76.6 ± 1.0 | 5.12 ± 0.29 |
| SketchGen | 89.4 ± 0.7 | 85.5 ± 0.8 | 79.1 ± 0.7 | 4.60 ± 0.23 |
| TransCAD 2024 | 91.0 ± 0.6 | 87.4 ± 0.7 | 84.8 ± 0.6 | 4.21 ± 0.19 |
| CADCodeVerify 2025 | 92.6 ± 0.5 | 88.7 ± 0.6 | 83.5 ± 0.7 | 4.06 ± 0.18 |
| CAD2Program 2025 | 91.1 ± 0.7 | 86.3 ± 0.9 | 82.2 ± 0.8 | 4.52 ± 0.31 |
| Drawing2CAD 2025 | 89.7 ± 0.8 | 85.7 ± 0.9 | 81.4 ± 0.9 | 4.43 ± 0.24 |
| Aligning Constraint Generation with Design Intent 2025 | 91.5 ± 0.6 | 88.3 ± 0.7 | 84.0 ± 0.6 | 4.12 ± 0.20 |
| FlexCAD | 91.9 ± 0.5 | 89.2 ± 0.6 | 84.2 ± 0.6 | 3.96 ± 0.18 |
| Proposed method | 92.0 ± 0.5 | 90.6 ± 0.5 | 84.5 ± 0.6 | 3.43 ± 0.16 |

As shown in Table 5, the proposed method achieves the highest CSR of 90.6% and the lowest AIC of 3.43 on DeepCAD. FlexCAD provides the strongest overall LLM-based comparison, reaching 89.2% CSR and 3.96 AIC, while CADCodeVerify obtains the highest CVR and TransCAD the highest EAA. Compared with FlexCAD, the proposed method improves CSR by 1.4 percentage points and reduces AIC by 0.53 iterations. These results indicate that its main advantage lies in constraint-aware iterative correction rather than uniform superiority in single-step validity or action accuracy.

Table 6. Main experimental results on SketchGraphs

| Method | CVR ↑ | CSR ↑ | EAA ↑ | AIC ↓ |
|---|---|---|---|---|
| Fusion 360 Gallery | 88.7 ± 0.8 | 83.0 ± 1.0 | 76.5 ± 0.9 | 4.91 ± 0.25 |
| Sketch2CAD | 87.8 ± 0.9 | 82.0 ± 1.1 | 76.4 ± 1.0 | 5.00 ± 0.28 |
| Free2CAD | 86.5 ± 1.0 | 80.5 ± 1.2 | 75.2 ± 1.1 | 5.22 ± 0.31 |
| SketchGen | 88.9 ± 0.8 | 85.3 ± 1.0 | 78.4 ± 0.8 | 4.71 ± 0.24 |
| TransCAD 2024 | 90.4 ± 0.6 | 86.7 ± 0.8 | 82.8 ± 0.7 | 4.30 ± 0.21 |
| CADCodeVerify 2025 | 91.8 ± 0.6 | 87.4 ± 0.7 | 81.7 ± 0.8 | 4.16 ± 0.19 |
| CAD2Program 2025 | 89.6 ± 0.8 | 85.4 ± 0.8 | 80.7 ± 0.8 | 4.39 ± 0.21 |
| Drawing2CAD 2025 | 88.4 ± 0.8 | 84.2 ± 0.9 | 79.6 ± 0.9 | 4.60 ± 0.24 |
| Aligning Constraint Generation with Design Intent 2025 | 91.2 ± 0.6 | 88.1 ± 0.7 | 82.4 ± 0.7 | 4.11 ± 0.20 |
| FlexCAD | 91.3 ± 0.6 | 88.5 ± 0.7 | 82.3 ± 0.7 | 4.02 ± 0.19 |
| Proposed method | 91.5 ± 0.5 | 89.4 ± 0.6 | 82.6 ± 0.6 | 3.61 ± 0.17 |

Table 6 shows a similar pattern on SketchGraphs. The proposed method obtains the highest CSR of 89.4% and the lowest AIC of 3.61. FlexCAD is the closest baseline, with 88.5% CSR and 4.02 AIC. The proposed framework therefore improves CSR by 0.9 percentage points and reduces AIC by 0.41 iterations relative to the LLM-based baseline, while maintaining comparable CVR and EAA.

Figure 4 summarizes the normalized scores and average metric ranks across both datasets. After including FlexCAD, the proposed method retains the strongest overall balance, primarily because of its higher CSR and lower AIC. FlexCAD performs competitively in CVR and EAA, whereas the proposed framework shows a clearer advantage in iterative constraint-aware refinement.

Figure 5 compares the iterative CSR trajectories of the proposed method with representative strong baselines, including TransCAD, CADCodeVerify, FlexCAD, and Aligning Constraint Generation with Design Intent. The proposed method reaches a higher constraint-satisfaction level within fewer refinement rounds, whereas the comparison methods require more iterations to reach the stopping condition or stabilize at lower CSR levels.

Figure 6 illustrates a representative feedback-driven CAD refinement case involving wall-thickness adjustment, hole modification, assembly-clearance control, and constraint checking.

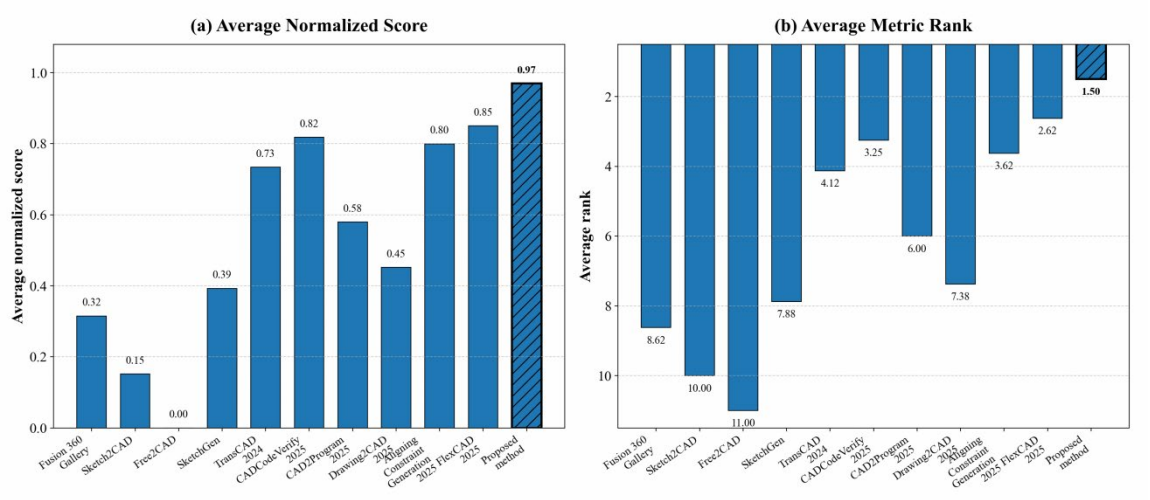


**Figure 4.** Overall performance of the CAD refinement benchmark
(a) Normalized score (b) Ranking of indicators

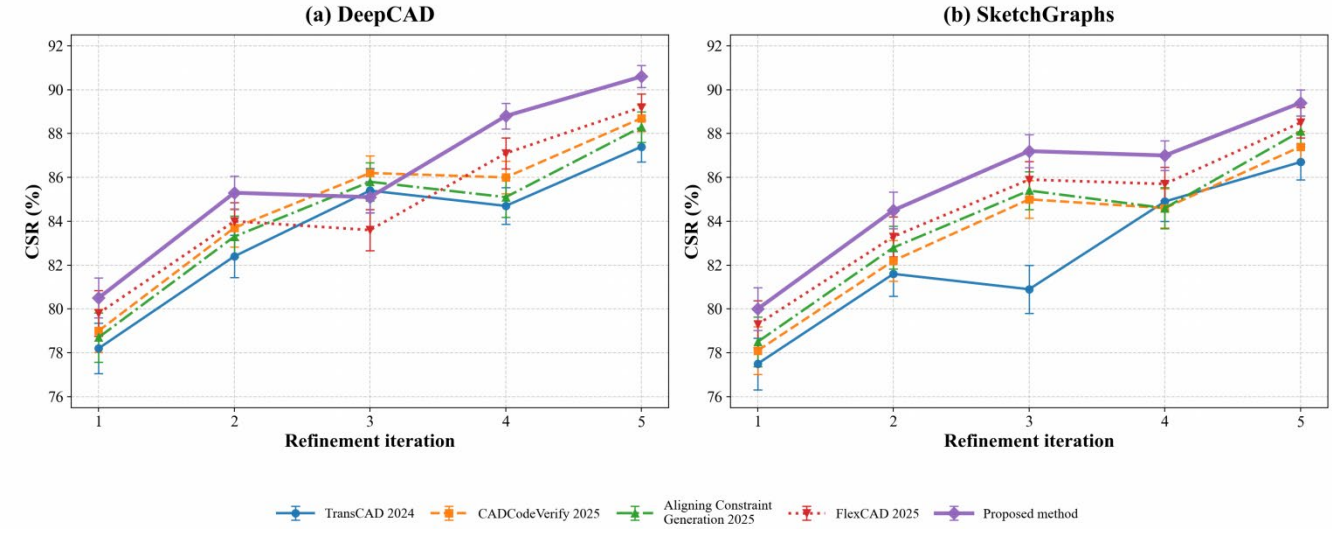


**Figure 5.** Changes in constraint satisfaction during the iterative CAD refinement process
(a) Iterative CSR curve on DeepCAD (b) Iterative CSR curve on SketchGraphs

**Figure 6.** Feedback-driven CAD refinement application

## 4.3. Statistical Significance Analysis

Paired t-tests were conducted against three strong baselines: CADCodeVerify 2025, Aligning Constraint Generation with Design Intent 2025, and FlexCAD 2025. Because this study focuses on constraint satisfaction and iterative efficiency, Table 7 reports CSR and AIC. Here, Δ denotes proposed minus baseline for CSR and baseline minus proposed for AIC.

As shown in Table 7, the proposed method achieves consistent improvements on both datasets. On DeepCAD, CSR exceeds the three baselines by 1.40–2.30 percentage points, with all p-values below 0.05, while AIC decreases by 0.53–0.69 iterations with all p-values below 0.01. On SketchGraphs, CSR improves by 0.90–2.00 percentage points and AIC decreases by 0.41–0.55 iterations. All 95% confidence intervals remain above zero, indicating consistent improvements across the five random-seed runs. These results are consistent with Tables 5 and 6, confirming that the main advantages of the proposed method lie in higher constraint satisfaction and lower iterative cost.

Figure 7 presents the corresponding 95% confidence intervals. The intervals remain above zero for both CSR improvement and AIC reduction, including the comparison with the LLM-based FlexCAD baseline, supporting the statistical significance of the observed gains.

Table 7. Paired t-test and 95% confidence interval analysis of core indicators

| Dataset | Metric | Compared Method | Δ | t-value | p-value | 95% CI |
|---|---|---|---|---|---|---|
| DeepCAD | CSR ↑ | CADCodeVerify 2025 | 1.90 | 4.91 | 0.008 | [0.83, 2.97] |
| DeepCAD | CSR ↑ | Aligning Constraint Generation with Design Intent 2025 | 2.30 | 4.18 | 0.014 | [0.77, 3.83] |
| DeepCAD | CSR ↑ | FlexCAD 2025 | 1.40 | 4.02 | 0.016 | [0.43, 2.37] |
| DeepCAD | AIC ↓ | CADCodeVerify 2025 | 0.63 | 6.28 | 0.003 | [0.35, 0.91] |
| DeepCAD | AIC ↓ | Aligning Constraint Generation with Design Intent 2025 | 0.69 | 6.75 | 0.002 | [0.41, 0.97] |
| DeepCAD | AIC ↓ | FlexCAD 2025 | 0.53 | 5.54 | 0.005 | [0.26, 0.80] |
| SketchGraphs | CSR ↑ | CADCodeVerify 2025 | 2.00 | 4.76 | 0.009 | [0.83, 3.17] |
| SketchGraphs | CSR ↑ | Aligning Constraint Generation with Design Intent 2025 | 1.30 | 3.62 | 0.022 | [0.30, 2.30] |
| SketchGraphs | CSR ↑ | FlexCAD 2025 | 0.90 | 3.31 | 0.030 | [0.15, 1.65] |
| SketchGraphs | AIC ↓ | CADCodeVerify 2025 | 0.55 | 5.67 | 0.005 | [0.28, 0.82] |
| SketchGraphs | AIC ↓ | Aligning Constraint Generation with Design Intent 2025 | 0.50 | 4.83 | 0.008 | [0.21, 0.79] |
| SketchGraphs | AIC ↓ | FlexCAD 2025 | 0.41 | 4.65 | 0.010 | [0.17, 0.65] |

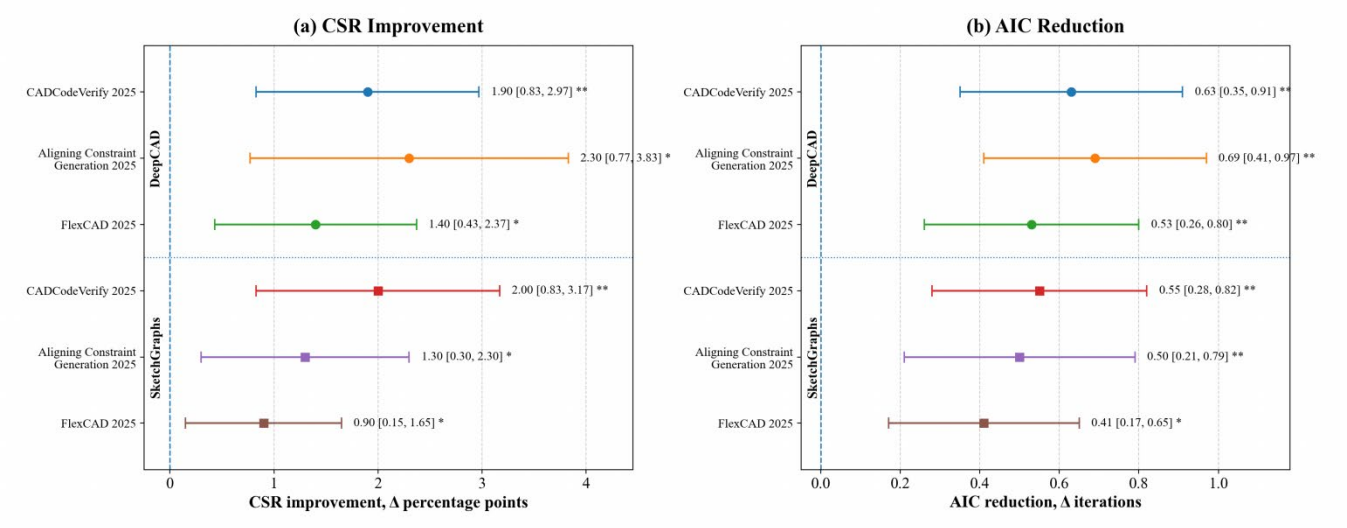


**Figure 7.** Confidence interval forest plot for performance improvement
(a) The improvement of CSR compared to the three strongest baselines
(b) The decrease of AIC compared to the three strongest baselines

## 4.4. Representation and Feature Space Analysis

To examine how CAD states, feedback information, and edit actions are organized in the learned feature space, latent representations from the trainable CAD-state representation and edit-action prediction network are analyzed on the same test splits using identical sampling settings and random seeds. The latent representations analyzed in this section are obtained from the trainable CAD-state representation and edit-action prediction network rather than from the fixed TF-IDF grounding module described in Section 3.3. The TF-IDF module is used only for deterministic feedback grounding and is not the source of the learned embeddings visualized below.

Figure 8 presents t-SNE visualizations of the learned CAD-state and feedback-related representations. Relative to the strongest baseline, the proposed method produces more clearly separated clusters for different edit operations and constraint states on both DeepCAD and SketchGraphs. The separation is particularly evident for samples involving coordinated design and constraint feedback. Some overlap remains for mixed-feedback samples, indicating that ambiguous or interacting feedback conditions are still more difficult to distinguish.

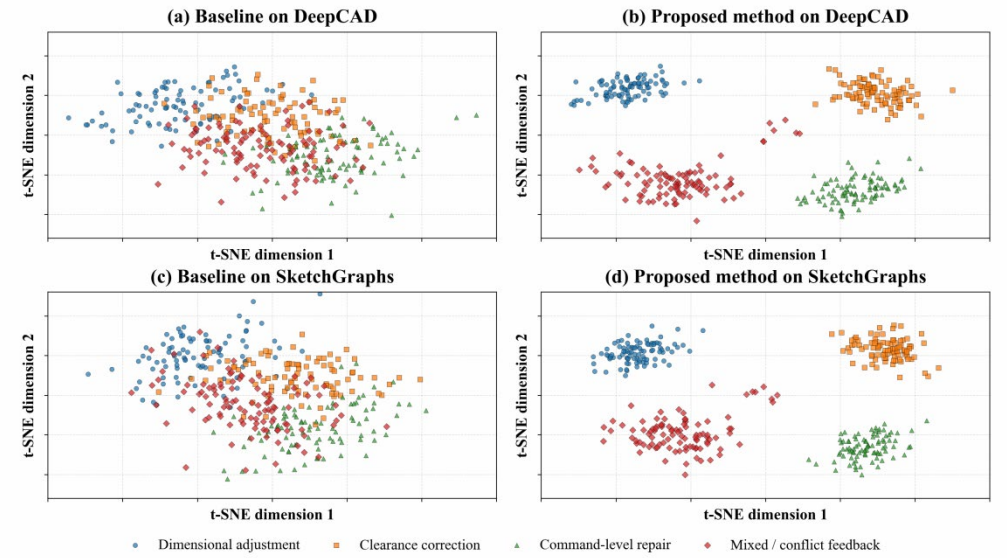

**Figure 8.** t-SNE visualization of learned CAD-state representations
(a) Baseline on DeepCAD (b) Proposed method on DeepCAD
(c) Baseline on SketchGraphs (d) Proposed method on SketchGraphs

Figure 9 compares feature distributions under four conditions: design feedback only, constraint feedback only, conflicting feedback, and coordinated dual feedback. The largest difference occurs under conflicting feedback, where the proposed method yields a more concentrated feature distribution. Differences are more visible in constraint-related confidence scores than in embedding norms, suggesting that conflict coordination primarily affects constraint-sensitive representations.

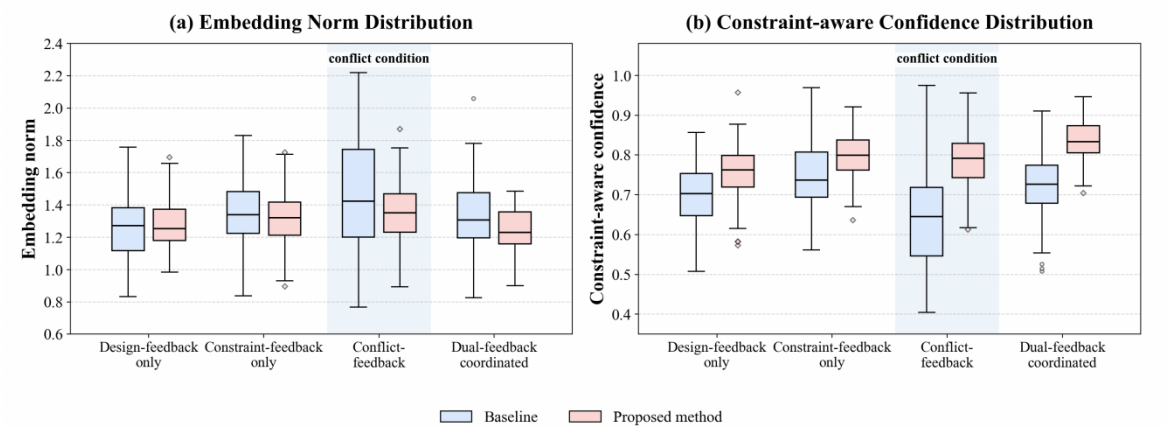


**Figure 9.** Feature distributions under four feedback conditions
(a) Embedding-norm distributions (b) Constraint-related confidence distributions

Figure 10 shows the similarity between learned feedback-related features and CAD-state embeddings. Higher similarities are observed between design-feedback cues and their associated CAD parameters, as well as between constraint-violation types and corresponding correction actions. These patterns indicate stronger alignment between learned representations of feedback information and editable CAD objects, local regions, and command-level actions.

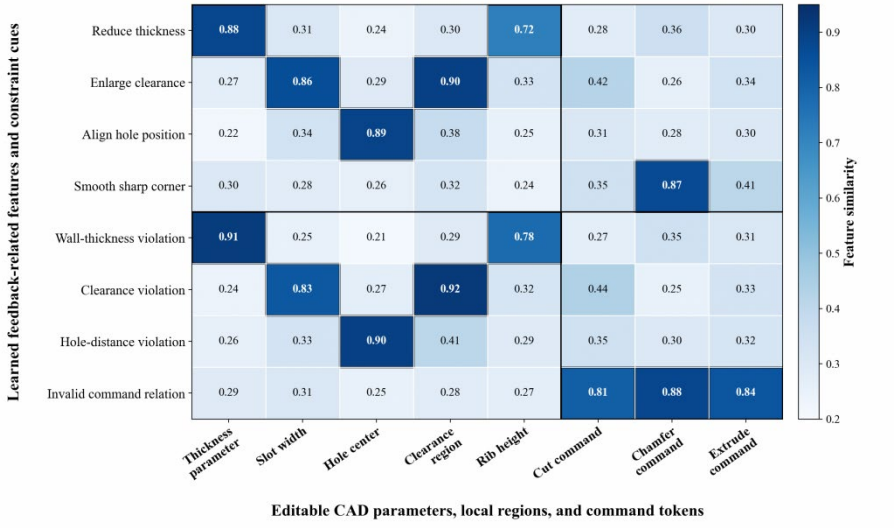


**Figure 10.** Similarity matrix between learned feedback-related features and CAD-state representations for editable parameters, constraint violations, and correction actions

## 4.5. Interpretability and Attention Analysis

To examine how the trainable prediction network uses CAD-state information and feedback signals during refinement, attention weights from the edit-action prediction branch are analyzed across editable parameters, local geometric regions, command tokens, and constraint-related cues. These attention weights originate from the trainable CAD-state representation and edit-action prediction network and are independent of the fixed TF-IDF grounding scores used in Section 3.3.

Figure 11 presents representative attention distributions associated with feedback-driven CAD edits. Relative to the baseline, the proposed method assigns larger weights to edit-relevant objects such as wall thickness, hole position, clearance regions, and command tokens associated with cutting, remodeling, chamfering, and filleting. When design and constraint feedback conflict, the attention distribution shifts toward the affected constraint objects and corresponding correction cues.

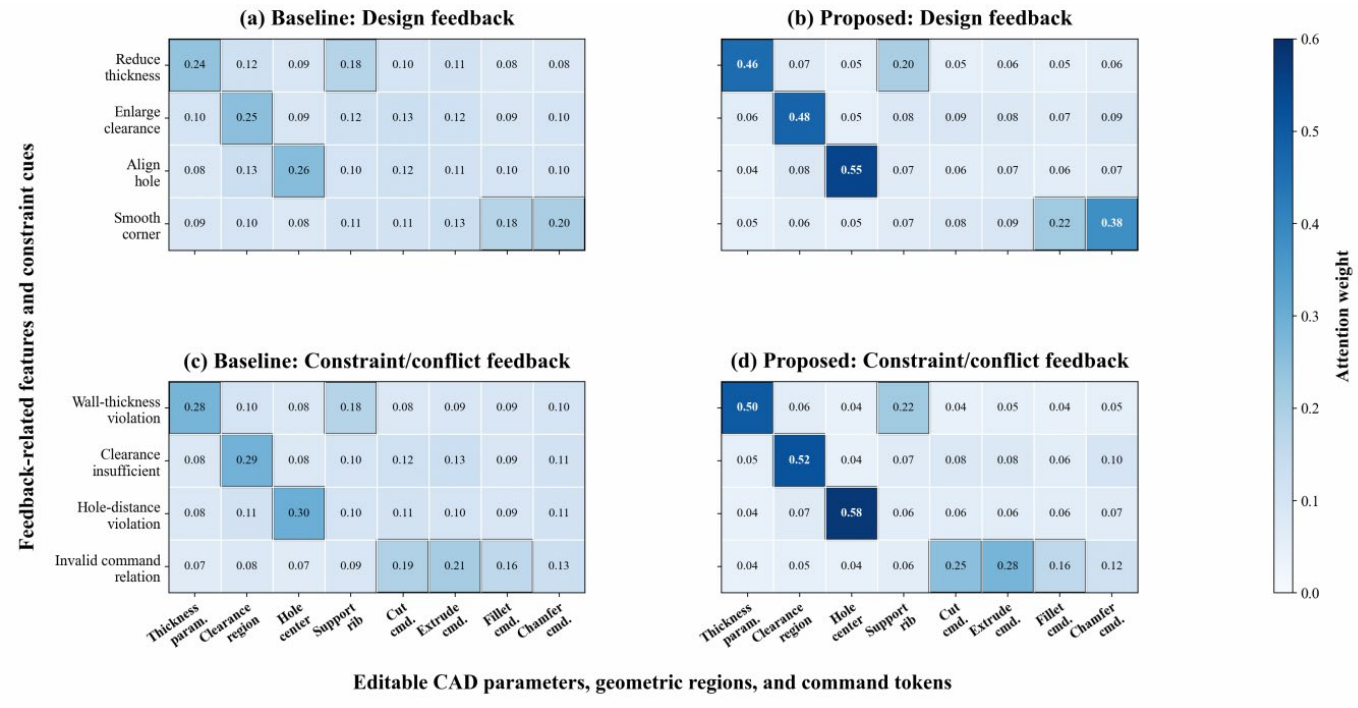


**Figure 11.** Attention distributions in the trainable edit-action prediction network over editable CAD parameters, geometric regions, and command tokens

Figure 12 shows the evolution of attention entropy during the five refinement iterations. The baseline exhibits larger fluctuations when design and constraint feedback interact. In contrast, the proposed method generally stabilizes after the initial refinement rounds, with attention becoming more concentrated on relevant parameters and constrained objects. Residual fluctuations remain in cases involving ambiguous feedback or simultaneously active constraints.

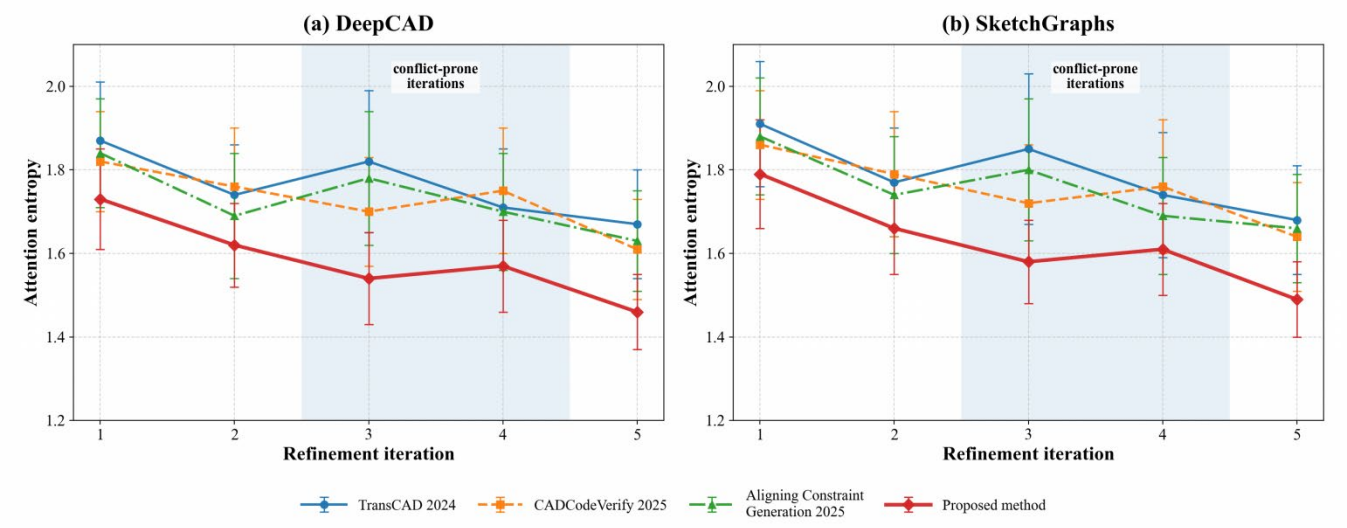


**Figure 12.** Attention-entropy trajectories across five refinement iterations (a) DeepCAD (b) SketchGraphs

Table 8 presents representative test cases linking attention allocation to predicted CAD edits. The attention score is calculated as the mean normalized weight across attention heads in the final attention layer of the trainable edit-action prediction branch. Highly weighted objects generally correspond to the predicted edit targets and CAD elements involved in subsequent constraint checks. Table 8 therefore complements Figures 11 and 12 by showing attention allocation at the parameter, local-region, and command levels.

Table 8. Representative Attention-aligned CAD editing cases

| Case ID | Dataset | Design Feedback | Constraint Feedback | Highest-attended CAD Object | Attention Score | Predicted Edit Action | Result |
|---|---|---|---|---|---|---|---|
| D-041 | DeepCAD | Reduce cover thickness | Minimum wall thickness violated | Top-cover thickness | 0.42 | Increase thickness slightly | Valid, constraint satisfied |
| D-087 | DeepCAD | Enlarge cable clearance | Clearance still insufficient | Inner clearance region | 0.39 | Expand slot width | Valid, constraint satisfied |
| S-126 | SketchGraphs | Make structure lighter | Thickness warning | Support rib | 0.36 | Reduce rib height moderately | Valid, minor constraint margin |
| S-214 | SketchGraphs | Align hole position | Hole distance violation | Hole center parameter | 0.44 | Shift hole center | Valid, constraint satisfied |
| D-193 | DeepCAD | Smooth sharp corner | Invalid command relation | Chamfer command token | 0.38 | Replace chamfer order | Valid after correction |

## 4.6. Error and Failure Case Analysis

Although the proposed method improves constraint satisfaction and iterative efficiency, several failure modes remain. Failed samples are identified from the DeepCAD and SketchGraphs test sets when the final CAD state is invalid, violates key constraints, or fails to match the target edit action within $T_{max} = 5$ refinement iterations.

Figure 13 summarizes the major failure types and their iterative characteristics. Ambiguous-feedback cases often fail because the target parameter or local region is insufficiently specified. Severe design–constraint conflicts generally require more refinement rounds and exhibit larger update fluctuations. Command-level failures mainly arise from unsupported or compound dependencies and invalid predicted operation sequences. Repeated oscillation near constraint boundaries remains less frequent but increases editing instability.

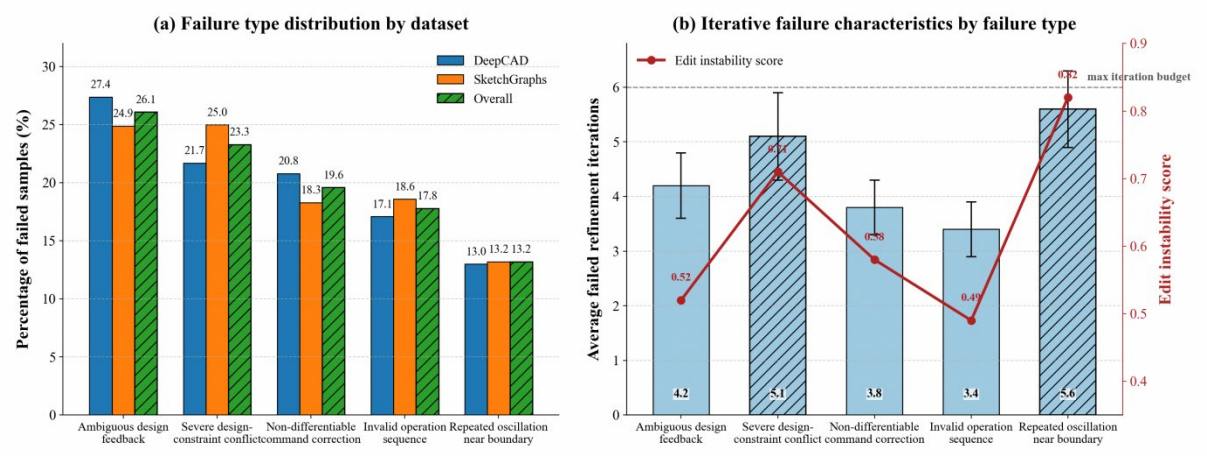


**Figure 13.** Failure-type distribution and iterative characteristics on DeepCAD and SketchGraphs
(a) Proportions of the five major failure types
(b) Average refinement iterations and editing-instability scores for each failure type

Table 9 summarizes the corresponding failure statistics. Ambiguous design feedback and severe design–constraint conflicts account for the largest proportions. Importantly, non-differentiable operations are not inherently treated as failures because the fixed correction-rule library $R_V$ handles supported discrete command dependencies. The remaining command-related failures primarily occur when compound or previously unsupported dependencies exceed the coverage of $R_V$, or when the predicted edit sequence violates CAD construction order.

A representative command-level correction is case D-193. The design feedback requests smoothing a sharp corner, but the initial operation sequence places a chamfer before its required parent extrusion. Because command order is discrete, a gradient cue cannot directly provide a valid reordering direction. The fixed rule library $R_V$, identifies the parent-operation dependency and moves the chamfer after the required extrusion while retaining the associated geometric parameters. Revalidation then confirms that the corrected operation sequence produces a valid CAD state. This case illustrates how rule-based write-back complements differentiable correction for supported discrete dependencies.

Table 9. Failure Type Statistics and Cause Analysis

| Failure Type | Percentage | Main Cause | Related Module | Possible Improvement |
|---|---|---|---|---|
| Ambiguous design feedback | 26.1% | Target parameter or local region is unclear | Design-feedback translation | Feedback disambiguation |
| Severe design–constraint conflict | 23.3% | Preference direction conflicts with active constraints | Conflict-aware coordination | Stronger conflict-resolution policy |
| Unsupported or compound command dependency | 19.6% | Dependency exceeds the coverage of the fixed correction-rule library | Constraint-feedback write-back | Expanded dependency rules or learned command-reasoning mechanism |

| | | | | |
|---|---|---|---|---|
| Invalid operation sequence | 17.8% | Predicted command breaks CAD construction order | Edit-action prediction | Operation-dependency modeling |
| Repeated oscillation near constraint boundary | 13.2% | Small reversed edits occur across iterations | History-aware damping | Adaptive edit-magnitude control |

## 4.7. Ablation Study

Ablation experiments were conducted on DeepCAD and SketchGraphs to evaluate design-feedback translation, constraint-feedback write-back, conflict-aware coordination, history-aware damping, and joint optimization of the trainable components. The fixed TF-IDF grounding, rule libraries, and coordination rules are not jointly trained. In the "w/o joint training" setting, only the trainable CAD-state representation and parameter/operation prediction branches are not jointly fine-tuned.

As shown in Table 10, removing design-feedback translation reduces EAA by 2.7 and 3.1 percentage points on DeepCAD and SketchGraphs, respectively. Removing constraint-feedback write-back causes larger CSR decreases, from 90.6% to 87.2% and from 89.4% to 85.6%. This supports violation-to-correction write-back rather than treating constraint feedback only as an external feasibility score.

Table 10. Ablation Study on DeepCAD and SketchGraphs

| Variant | DeepCAD CVR ↑ | DeepCAD CSR ↑ | DeepCAD EAA ↑ | DeepCAD AIC ↓ | SketchGraphs CVR ↑ | SketchGraphs CSR ↑ | SketchGraphs EAA ↑ | SketchGraphs AIC ↓ |
|---|---|---|---|---|---|---|---|---|
| Full model | 92.0 ± 0.5 | 90.6 ± 0.5 | 84.5 ± 0.6 | 3.43 ± 0.16 | 91.5 ± 0.5 | 89.4 ± 0.6 | 82.6 ± 0.6 | 3.61 ± 0.17 |
| w/o design-feedback translation | 91.5 ± 0.6 | 89.0 ± 0.8 | 81.8 ± 0.8 | 3.86 ± 0.22 | 90.9 ± 0.7 | 87.9 ± 0.8 | 79.5 ± 0.9 | 4.07 ± 0.25 |
| w/o constraint-feedback write-back | 91.1 ± 0.7 | 87.2 ± 0.9 | 83.6 ± 0.7 | 3.98 ± 0.24 | 90.6 ± 0.8 | 85.6 ± 1.0 | 81.5 ± 0.8 | 4.25 ± 0.27 |
| w/o conflict-aware coordination | 91.3 ± 0.6 | 87.7 ± 0.8 | 82.9 ± 0.8 | 4.31 ± 0.27 | 90.5 ± 0.7 | 86.6 ± 0.9 | 80.8 ± 0.9 | 4.56 ± 0.30 |
| w/o history-aware damping | 92.0 ± 0.6 | 89.4 ± 0.7 | 84.1 ± 0.6 | 4.02 ± 0.22 | 91.6 ± 0.7 | 88.1 ± 0.8 | 81.9 ± 0.7 | 4.29 ± 0.26 |
| w/o joint training | 91.4 ± 0.6 | 88.4 ± 0.7 | 82.8 ± 0.8 | 3.93 ± 0.21 | 90.9 ± 0.6 | 87.3 ± 0.8 | 81.2 ± 0.8 | 4.09 ± 0.24 |
| design-feedback only | 90.8 ± 0.8 | 86.0 ± 1.0 | 83.8 ± 0.7 | 4.26 ± 0.26 | 90.0 ± 0.9 | 85.1 ± 1.1 | 81.5 ± 0.8 | 4.46 ± 0.29 |
| constraint-feedback only | 91.1 ± 0.7 | 88.2 ± 0.8 | 80.7 ± 1.0 | 3.91 ± 0.23 | 90.4 ± 0.8 | 87.0 ± 0.9 | 79.0 ± 1.1 | 4.14 ± 0.27 |

Conflict-aware coordination affects iterative decision behavior rather than merely combining two feedback sources. Without it, AIC rises from 3.43 to 4.31 on DeepCAD and from 3.61 to 4.56 on SketchGraphs, the largest increases among the module-removal settings. CSR and EAA also decline, indicating that coordination directly influences edit selection across refinement rounds.

History-aware damping provides an independent benefit despite its low-complexity design. Removing it increases AIC from 3.43 to 4.02 and from 3.61 to 4.29, while CVR changes only slightly. This supports the direction-change-count mechanism in Section 3.5, whose main role is to suppress repeated reversals and unnecessary edits.

The single-feedback settings further show that the two feedback sources are complementary. Design-feedback-only refinement retains relatively high EAA but lowers CSR, whereas constraint-feedback-only refinement improves CSR relative to the design-only setting but reduces EAA. Removing joint training also degrades all metrics; here, joint training refers only to the trainable CAD-state representation and edit-action prediction branches.

Figure 14 summarizes these effects. Removing design-feedback translation reduces EAA by 2.7 and 3.1 points, while removing constraint-feedback write-back reduces CSR by 3.4 and 3.8 points. Conflict-aware coordination causes the largest AIC increase, followed by history-aware damping and joint training. Sensitivity results further show the strongest performance near the default coordination and damping settings.

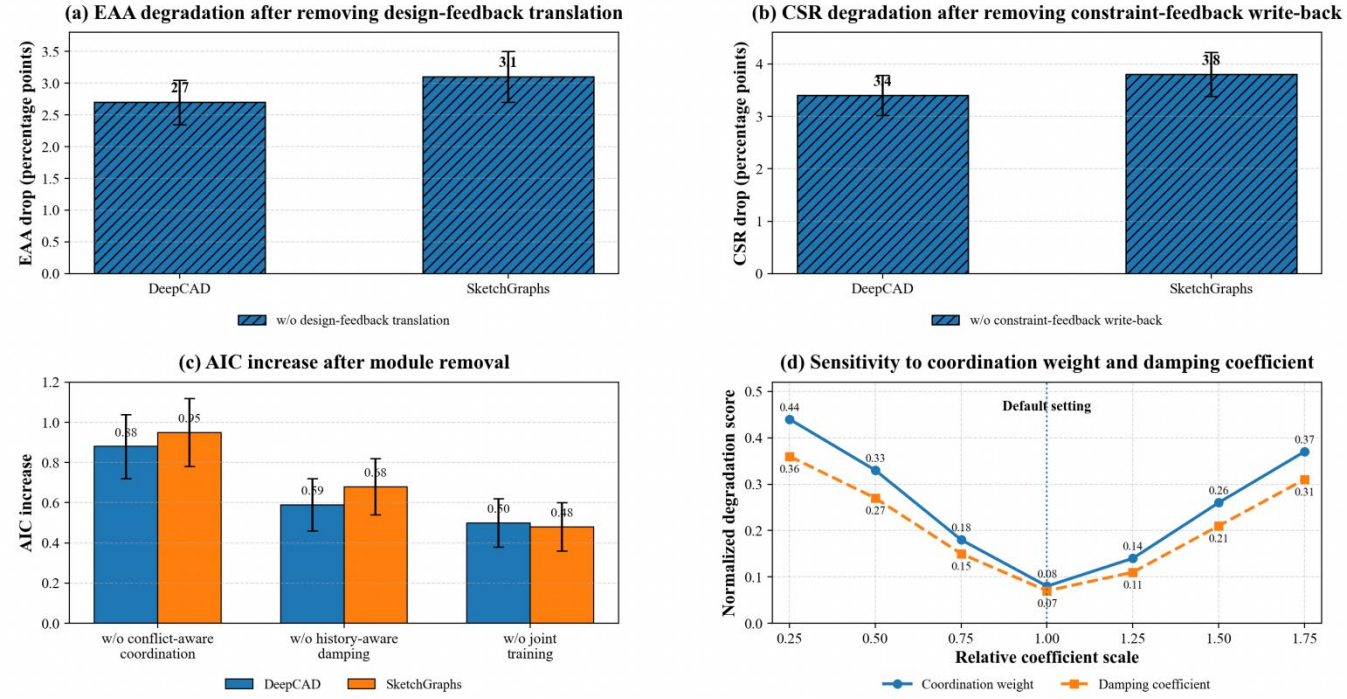


**Figure 14.** Ablation and sensitivity analysis of the proposed framework
(a) EAA degradation after removing design-feedback translation
(b) CSR degradation after removing constraint-feedback write-back
(c) AIC increase after removing conflict-aware coordination, history-aware damping, or joint training
(d) Sensitivity to relative scaling of the coordination weight and damping coefficient

## 5. Discussion and Conclusion

### 5.1. Summary of the Study

The experimental results show that the main advantage of the proposed framework lies in constraint-aware iterative refinement rather than uniformly maximizing all CAD-generation metrics. On DeepCAD and SketchGraphs, it achieves CSR values of 90.6% and 89.4% with AIC values of 3.43 and 3.61, respectively, while maintaining competitive CVR and EAA. The gains in CSR and AIC indicate that feedback is used to guide subsequent corrections and reduce unnecessary refinement rounds. In contrast, several one-shot, LLM-based, and verification-oriented CAD methods remain competitive in validity or edit accuracy but provide less support for repeated feedback-driven correction. The results therefore suggest that closed-loop refinement is particularly useful when design preferences and engineering constraints must be reconciled across multiple edit steps.

## 5.2. Research Limitations

Several limitations remain. First, the feedback signals are reconstructed from DeepCAD and SketchGraphs rather than collected from complete industrial revision sessions. This limits ecological validity, and the current results cannot directly establish industrial effectiveness. No human-participant pilot or complete industrial CAD revision study is included in the present evaluation; therefore, conclusions regarding designer acceptance and real-workflow effectiveness remain outside the scope of this study. Real design feedback may be incomplete, ambiguous, delayed, evolving, or supplied by multiple designers with inconsistent preferences. Raw image-based and multimodal designer feedback are also outside the current grounding pipeline and require dedicated visual-semantic encoding in future work.

Second, the current constraint set mainly covers geometric, dimensional, assembly, volume, topology, and command-related conditions. Structural strength, thermal behavior, material properties, manufacturability, and coupled physical effects are not comprehensively represented. Scalability to larger assemblies, multiple simultaneously active constraints, and heterogeneous feedback sources also requires further evaluation. The current experiments do not constitute a dedicated assembly-scale stress test, so computational and decision complexity under substantially larger CAD models remains to be established.

Third, the current conflict mechanism mainly handles local directional, range-overlap, and priority-based conflicts. Nonlinear dependencies among multiple objects, continuous preference ranges, and long-horizon command dependencies remain difficult to resolve. History-aware damping reduces repeated reversals but does not fully model long-range constraint propagation. More general coordination may therefore require multi-criteria decision making, learning-to-rank, or adaptive control mechanisms.

## 5.3. Future Development

Future work should construct industrial feedback datasets containing authentic CAD revision logs, natural-language designer comments, simulation-violation records, and complete multi-round design sessions. Because such industrial revision data are not included in the present evaluation, the current results should be interpreted as benchmark-level validation rather than direct evidence of industrial deployment effectiveness. Future evaluation will re-run the core benchmarks on authentic ambiguous and multi-designer feedback once such data become available. Such datasets should include incomplete requirements, evolving preferences, and conflicting feedback from multiple designers. Future extensions could retain unresolved edit targets when feedback is incomplete, introduce confidence-based disambiguation or designer confirmation for ambiguous instructions, and associate delayed feedback with the corresponding versioned CAD state before correction write-back. Human-in-the-loop studies could further evaluate editing efficiency, designer acceptance, and decision consistency under realistic design conditions.

A second direction is to extend constraint feedback toward physics-based and multi-physics validation. Structural strength, thermal behavior, material constraints, manufacturability, and assembly performance could be connected through simulation interfaces and constraint-propagation models. This would allow correction cues to reflect not only geometric feasibility but also downstream engineering performance.

Finally, practical deployment requires tighter integration with CAD and product-lifecycle infrastructure. In an engineering workflow, designer input and captured feedback would first be processed by the grounding and refinement engine, after which the resulting edit instructions would be executed through a CAD API or geometric kernel. The updated CAD state could then be evaluated by simulation or constraint-checking tools before being written back to the CAD environment and recorded in a PLM or version-control system. Integration through the Fusion 360 API, SolidWorks API, commercial CAD kernels, simulation-solver interfaces, and PLM systems could support this closed-loop workflow. Practical deployment must additionally address CAD-command latency, solver communication, version rollback, designer approval, and access control.

**Acknowledgements.**
This study was supported by Special Project Funding from the Research Center for the Development of New Quality Productive Forces in Tourism of Nanchong City, "Research on Accelerating the Construction of a Cultural Tourism Development Demonstration Zone with Bashu Characteristics in Nanchong" (Grant No. WLXZ2025B017) and Sichuan Technology and Business University University-level Scientific Research Project (2025, Grant No. XJ2025YB031)

## References

[1] Khan, M. S., Sinha, S., Sheikh, T. U., Stricker, D., Ali, S. A., & Afzal, M. Z. (2024). Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. Advances in Neural Information Processing Systems, 37, 7552-7579.

[2] Li, X., Song, Y., Lou, Y., & Zhou, X. (2024, October). Cad translator: An effective drive for text to 3d parametric computer-aided design generative modeling. In Proceedings of the 32nd ACM International Conference on Multimedia (pp. 8461-8470).

[3] Liu, Y., Obukhov, A., Wegner, J. D., & Schindler, K. (2024). Point2cad: Reverse engineering cad models from 3d point clouds. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 3763-3772).

[4] Ma, W., Chen, S., Lou, Y., Li, X., & Zhou, X. (2024). Draw Step by Step: Reconstructing CAD Construction Sequences from Point Clouds via Multimodal Diffusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 27154-27163).

[5] Stauss, T., Wolniak, P., Tergeist, M., Wurst, J., & Lachmayer, R. (2025). Enhancing product design with digital twins: framework and application in an industry case study. Proceedings of the Design Society, 5, 1555-1564.

[6] Wang, P., Khinvasara, Y., Creijghton, G. J., Scholing, T., Wang, Y., Zhou, Z., Childs, P. R. N., & Yin, Y. (2025). Enhancing designer creativity through human–AI co-ideation: A co-creation framework for design ideation with custom GPT. Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 39, e22.

[7] Li, X., Lou, Y., Song, Y., & Zhou, X. (2025, April). Mamba-cad: State space model for 3d computer-aided design generative modeling. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 5, pp. 5013-5021).

[8] Alrashedy, K., Tambwekar, P., Zaidi, Z. H., Langwasser, M., Xu, W., & Gombolay, M. (2025, May). Generating cad code with vision-language models for 3d designs. In International Conference on Learning Representations (Vol. 2025, pp. 52236-52262).

[9] Li, J., Ma, W., Li, X., Lou, Y., Zhou, G., & Zhou, X. (2025). CAD-Llama: leveraging large language models for computer-aided design parametric 3D model generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 18563-18573).

[10] Liao, J., Xu, J., Sun, Y., Tang, M., He, S., Liao, J., Yu, S., Li, Y., & Guan, X. (2025). Automated CAD modeling sequence generation from text descriptions via transformer-based large language models. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 21720–21748)..

[11] Xu, X., Lambourne, J., Jayaraman, P., Wang, Z., Willis, K., & Furukawa, Y. (2024). Brepgen: A b-rep generative diffusion model with structured latent geometry. ACM Transactions on Graphics (TOG), 43(4), 1-14.

[12] Zhang, Z., Sun, S., Wang, W., Cai, D., & Bian, J. (2025, May). Flexcad: Unified and versatile controllable cad generation with fine-tuned large language models. In International Conference on Learning Representations (Vol. 2025, pp. 3204-3227).

[13] Deng, L., Bai, Y., Dai, Y., Huang, X., Gan, H., Huang, D., Jiacheng, H., & Shi, Y. (2025). MamTiff-CAD: Multi-scale latent diffusion with Mamba+ for complex parametric sequence. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 10517–10526).

[14] Li, P., Zhang, W., Guo, J., Chen, J., & Yan, D. M. (2025, April). Revisiting cad model generation by learning raster sketch. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 5, pp. 4869-4877).

[15] Li, Y., Lin, C., Liu, Y., Long, X., Zhang, C., Wang, N., Li, X., Wang, W., & Guo, X. (2025). CADDreamer: CAD object generation from single-view images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 21448–21457).

[16] Mallis, D., Karadeniz, A. S., Cavada, S., Rukhovich, D., Foteinopoulou, N., Cherenkova, K., Kacem, A., & Aouada, D. (2025). CAD-Assistant: Tool-augmented VLLMs as generic CAD task solvers. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 7284–7294).

[17] Lee, M., Zhang, D., Jambon, C., & Kim, Y. M. (2025, August). Brepdiff: Single-stage b-rep diffusion model. In Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (pp. 1-11).

[18] Liu, Y., Xu, D., Yu, X., Xu, X., Cohen-Or, D., Zhang, H., & Huang, H. (2025). HoLa: B-Rep generation using a holistic latent representation. ACM Transactions on Graphics, 44(4), 1–25.

[19] Casey, E., Zhang, T., Ishida, S., Thompson, J. R., Khasahmadi, A., Lambourne, J. G., Jayaraman, P. K., & Willis, K. D. D. (2025). Aligning constraint generation with design intent in parametric CAD. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 8613–8622).

[20] Zhou, J., Camba, J. D., & Company, P. (2026). CADialogue: A multimodal LLM-powered conversational assistant for intuitive parametric CAD modeling. Computer-Aided Design, 191, 104006.

[21] Wang, W. F., Lu, C. T., Ponsa i Campanyà, N., Chen, B. Y., & Chen, M. Y. (2025, April). AIdeation: Designing a human-AI collaborative ideation system for concept designers. In Proceedings of the 2025 chi conference on human factors in computing systems (pp. 1-28).

[22] Yao, J., Chen, P., Li, Z., Cai, Y., Wu, Y., You, W., & Sun, L. (2025). StepIdeator: Utilizing mixed representations to support step-by-step design with generative artificial intelligence. Journal of Mechanical Design, 147(7), 071703.

[23] Edwards, K. M., Man, B., & Ahmed, F. (2024). Sketch2Prototype: rapid conceptual design exploration and prototyping with generative AI. Proceedings of the Design Society, 4, 1989-1998.

[24] Koyama, Y., & Goto, M. (2022, October). BO as assistant: Using Bayesian optimization for asynchronously generating design suggestions. In Proceedings of the 35th annual ACM symposium on user interface software and technology (pp. 1-14).

[25] Nandy, A., & Goucher-Lambert, K. (2026). Exploring the effectiveness of interactive preference learning for adapting designs to abstract semantic attributes. Journal of Mechanical Design, 148(4), 041702.

[26] Anwer, N., Stark, R., Tao, F., & Erkoyuncu, J. A. (2025). Developing and leveraging digital twins in engineering design. CIRP Annals, 74(2), 843-868.

[27] Zhang, Y., Bertoni, M., Bertoni, A., Larsson, A., & Larsson, T. (2025). Leveraging digital twins for value-driven design in smart product-service systems: the super-system digital twin framework and SEV case study. Design Science, 11, e24.

[28] Mercat, F., Le Masson, P., & Weil, B. (2025). The unique feature of designing with digital twin uncovered by design theory. Proceedings of the Design Society, 5, 3101-3110.

[29] Machacek, Z., Hercik, R., Vaclavik, A., Zemanek, J., Hameed, I. A., & Koziorek, J. (2025). Modern trends and industrial use cases of digital twin technology with 3D behavioral representation. Journal of Intelligent Manufacturing, 1-34.

[30] Mata, O., Ponce, P., Perez, C., Ramirez, M., Anthony, B., Russel, B., Apte, P., MacCleery, B., & Molina, A. (2026). Digital twin designs with generative AI: Crafting a comprehensive framework for manufacturing systems. Journal of Intelligent Manufacturing. 37, 1049–1072.

[31] Sun, Z., Wang, Y., Guo, W., & Meng, Q. (2026). Human-in-the-Loop Digital Twin Modeling for Smart Civil Infrastructure Operation and Maintenance. Applied Sciences, 16(4), 1848.

[32] Ruzarovsky, R., Horak, T., Zelník, R., Skypala, R., Csekei, M., Šido, J., Nemlaha, E., & Kopcek, M. (2025). Development and validation of digital twin behavioural model for virtual commissioning of cyber-physical system. Applied Sciences, 15(5), 2859.

[33] Papacharalampopoulos, A., Sabatakakis, K., Karagianni, O. M., & Stavropoulos, P. (2024, October). Digital Twins of Manufacturing Processes Under Industry 5.0. In European Symposium on Artificial Intelligence in Manufacturing (pp. 3-11). Cham: Springer Nature Switzerland.

[34] Willis, K. D. D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J. G., Solar-Lezama, A., & Matusik, W. (2021). Fusion 360 Gallery: A dataset and environment for programmatic CAD construction from human design sequences. ACM Transactions on Graphics, 40(4), Article 54.

[35] Li, C., Pan, H., Bousseau, A., & Mitra, N. J. (2020). Sketch2cad: Sequential cad modeling by sketching in context. ACM Transactions on Graphics (TOG), 39(6), 1-14.

[36] Li, C., Pan, H., Bousseau, A., & Mitra, N. J. (2022). Free2cad: Parsing freehand drawings into cad commands. ACM Transactions on Graphics (TOG), 41(4), 1-16.

[37] Para W, Bhat S, Guerrero P, Kelly T, Mitra N, Guibas LJ, et al. (2021). Sketchgen: Generating constrained cad sketches. Advances in Neural Information Processing Systems, 34, 5077-5088.

[38] Dupont, E., Cherenkova, K., Mallis, D., Gusev, G., Kacem, A., & Aouada, D. (2024, September). Transcad: A hierarchical transformer for cad sequence inference from point clouds. In European Conference on Computer Vision (pp. 19-36). Cham: Springer Nature Switzerland.

[39] Wang, X., Zheng, J., Hu, Y., Zhu, H., Yu, Q., & Zhou, Z. (2025, April). From 2d cad drawings to 3d parametric models: A vision-language approach. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 8, pp. 7961-7969).

[40] Qin, F., Lu, S., Hou, J., Wang, C., Fang, M., & Liu, L. (2025, October). Drawing2CAD: Sequence-to-Sequence Learning for CAD Generation from Vector Drawings. In Proceedings of the 33rd ACM International Conference on Multimedia (pp. 10573-10582).