PROMPTABLE GEOMETRIC GROUNDING IN B-REP MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the task of grounding in scenes represented by discrete topology and continuous geometry, e.g., Boundary Representation (B-Rep) models. Unlike traditional 3D visual grounding that localizes objects in real-world point cloud scenes, the task requires both a fundamentally different scene representation and a finer localization granularity, where targets are not objects but topologically defined primitives (e.g., faces and edges) embedded in a structured geometric model. This paper introduces PGGM, a Promptable Geometric Grounding Model for B-Rep models. PGGM jointly supports natural language descriptions and partial selections as prompts. It decouples continuous geometry and discrete topology: pretrained VAEs encode geometric shape information, while a topology encoder captures structured primitive relations. These representations are then integrated through prompt-conditioned fusion and primitive-level decoding for fine-grained grounding. To further improve hard-negative discrimination, we introduce Discriminative Bootstrapping (DB), a training strategy that leverages the partial branch to turn noisy auxiliary signals into discriminative supervision. DB constructs complementary signals from external teachers, model self-bootstrapping, and proximity-based hard negatives, exposing the model to distractors that are semantically plausible, self-predicted, or close to the target in latent space. This enables more reliable grounding under fine-grained geometric ambiguity. Experiments show that PGGM significantly improves B-Rep grounding over strong baselines, achieving a +17.26% F1 gain and a +23.73% precision gain in the text-only setting.

1 INTRODUCTION

Computer-aided design (CAD) plays a central role in the representation and construction of engineered objects Cherng et al. (1998). In CAD modeling, users routinely ground modeling intent to concrete B-Rep primitives, such as selecting edges for a *fillet* operation Daareyni et al. (2025). Automating this grounding step is fundamental to intelligent interactive CAD modeling, where operations can be driven by intent rather than manual operand specification, yet it remains largely unexplored. Beyond interactive modeling, B-Rep grounding also complements parametric CAD generation Qi et al. (2026); Li et al. (2026), as most existing methods synthesize modeling histories yet lack the primitive-level grounding required for advanced features.

Although text-guided 3D visual grounding has seen rapid progress, with methods operating on point cloud scenes to localize objects via bounding box regression or instance matching Li et al. (2025b); Huang et al. (2025; 2026); Zemskova & Yudin (2024), these methods are designed for scenes where objects are geometrically separable, semantically nameable, and localized by spatial extent Wang et al. (2025). In contrast, B-Rep grounding fundamentally differs from this setting: the input is a structured topology of faces and edges rather than a point cloud, and the target is a set of topologically defined primitives that share no perceptual distinctiveness from their neighbors. B-Rep representation learning, on the other hand, has largely developed around tasks of shape classification Deng et al. (2025); Lambourne et al. (2021), machining feature recognition Dai et al. (2025); Zhang et al. (2024), and generative reconstruction Jayaraman et al. (2023); Xu et al. (2025). These methods learn to assign categorical labels or reconstruct geometric structure, neither of which requires grounding natural language into a selection of semantically relevant primitives. The task of B-Rep grounding sits at the intersection of these two lines of work, yet is addressed by neither.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This gap can be attributed to several challenges inherent to the fine-grained and structured nature of B-Rep grounding. *Geometry-topology coupling* requires the model to jointly interpret continuous shape attributes and discrete primitive relations, since grounding targets are often determined not by local geometry alone but by their role within neighboring faces, edges, and connectivity patterns. This further leads to *hard-negative ambiguity*: many non-target primitives are geometrically similar, topologically adjacent, or intent-relevant, making them difficult to distinguish from true targets. Meanwhile, *language-geometry misalignment* introduces a large representational gap, as natural descriptions provide semantic and spatial cues, while the target primitives must be identified through geometric attributes and topological context in the B-Rep. Together, B-Rep grounding represents a fundamentally distinct challenge that is not addressed by existing lines of work.

In this paper, we introduce PGGM, a Promptable Geometric Grounding Model for B-Rep models. PGGM separates continuous geometric representation from discrete topological reasoning to explicitly model *geometry-topology coupling*, allowing local shape evidence and structured primitive relations to be modeled in complementary branches. Pretrained face and edge VAEs Xu et al. (2024) provide stable geometric embeddings, while coedge-based message passing Lambourne et al. (2021) captures the connectivity structure of B-Rep primitives. The resulting representations are fused with prompt tokens in a geometry-aware manner and decoded into primitive-level grounding predictions. To exploit partial inputs when available, PGGM introduces Geometric Similarity Propagation (GSP), which summarizes selected primitives as a prototype and propagates similarity cues to related candidates. PGGM is further trained with contrastive objectives van den Oord et al. (2018) that enhance positive-negative separation and mitigate *language-geometry misalignment* by aligning prompt semantics with B-Rep geometry.

Hard-negative ambiguity remains a significant challenge for primitive-level discrimination. To improve robustness in such cases, we introduce Discriminative Bootstrapping (DB), a training strategy built on the partial branch. DB leverages noisy auxiliary signals through the partial branch, training the model to distinguish useful cues from hard-negative noise in imperfect partial inputs. The signals are constructed from three complementary sources: Teacher-Induced Discrimination (TID) for external semantic ambiguity, Self-Bootstrapped Discrimination (SBD) for model-induced ambiguity, and Proximity-Contrastive Discrimination (PCD) for latent-space ambiguity. This design turns noisy partial signals into supervision for learning more discriminative primitive representations, which in turn improves the model's ability to distinguish hard negatives and alleviates *language-geometry misalignment*, enabling more fine-grained and semantically faithful grounding.

Experiments show that PGGM substantially improves primitive-level grounding on B-Rep models, consistently outperforming baselines with strong gains. We also validate its effectiveness in practical modeling scenarios by integrating it into parametric CAD generation frameworks, as detailed in Appendix B. Our contributions are summarized as follows:

- We introduce PGGM, a Promptable Geometric Grounding Model for B-Rep models that explicitly models geometry-topology coupling via separate geometric and topological branches, and unifies textual prompts and partial inputs for multi-conditional primitive-level prediction.
- We introduce Discriminative Bootstrapping (DB), a general training strategy for improving hard-negative discrimination through bootstrapped conditional signals, improving robustness to teacher-induced, self-induced, and proximity-based hard-negative ambiguity.
- Experiments show that PGGM significantly advances B-Rep grounding, yielding +17.26% F1 and +23.73% precision improvements over strong baselines in the text-only setting.

2 Related Work

B-Rep Representation Learning. Recent B-Rep learning methods focus on jointly modeling geometry and topology in native CAD representations. Early methods such as BRepNet directly perform message passing over coedge-centered B-Rep neighborhoods, while UV-Net unifies parametric surface and curve representations with adjacency-based topology through a hybrid CNN-GNN design Lambourne et al. (2021); Jayaraman et al. (2021). Subsequent work improves representation capacity with stronger architectures: BRepFormer Dai et al. (2025) and BrepMFR Zhang et al. (2024) use Transformer-based graph reasoning, VGNet wei Qin et al. (2025) explores multimodal fusion, and FilletRec Gao et al. (2025) shows that lightweight graph models with intrinsic geometric features can achieve strong performance. In parallel, self-supervised learning reduces annotation dependence,

with Self-Supervised Representation Learning for CAD Jones et al. (2023) and BRepMAE Yao et al. (2026) showing that pretraining on unlabeled B-Rep data improves transferability and data efficiency. Beyond analysis tasks, recent generative models further broaden B-Rep learning by directly synthesizing valid CAD representations using autoregressive, diffusion, and latent-space approaches Jayaraman et al. (2023); Xu et al. (2024); Li et al. (2025a); Liu et al. (2025); Xu et al. (2025). However, most existing methods focus on representation learning and generation, without explicitly addressing grounding scenarios where models must localize and select relevant B-Rep primitives under varying conditions. While FutureCAD Li et al. (2026) takes a first step toward text-based grounding via LLM-driven program generation, it operates with text-only queries in a sequential pipeline and does not support multi-condition inputs or robust entity-level selection under practical CAD interactions.

Furthermore, existing B-Rep methods are trained under clean supervision and lack mechanisms to learn discriminative representations against geometrically entangled hard negatives, limiting robust geometric grounding. While SAM Kirillov et al. (2023) introduces noisy prompts, its i.i.d. positional perturbations mainly improve prompt robustness rather than discrimination. In contrast, we propose Discriminative Bootstrapping (DB), a training strategy that explicitly improves discriminative learning by exposing the model to external, self-induced, and representation-near distractors as noisy conditional inputs.

3D Visual Grounding. 3D visual grounding aims to localize target entities in 3D scenes from natural language descriptions. Early benchmarks such as ScanRefer Chen et al. (2019) and ReferIt3D Achlioptas et al. (2020) establish this task in RGB-D scans and point-cloud scenes, prompting methods that learn joint language-geometry representations for object localization. Subsequent work improves grounding through stronger contextual modeling, for example by incorporating relation-aware reasoning over object proposals or multi-view representations to better resolve ambiguous spatial descriptions Zhao et al. (2021); Huang et al. (2022). More recent studies further extend the setting toward open-vocabulary and reasoning-based grounding by leveraging large language models or vision-language models Yang et al. (2024); Li et al. (2025c), while benchmarks such as Anywhere3D-Bench Wang et al. (2025) show that grounding beyond object level, such as free space or object parts, remains highly challenging. Despite these advances, existing 3D visual grounding methods are primarily designed for real-world scene understanding in point-cloud environments, where the goal is to localize objects in continuous 3D space. In contrast, our setting targets CAD B-Rep models and requires precise grounding of discrete geometric primitives. This makes the task fundamentally different and often more challenging, as models must resolve fine-grained geometric and topological ambiguities rather than coarse object-level localization. As a result, methods developed for object grounding in real-world point-cloud scenes are not directly applicable to structured B-Rep grounding.

3 METHOD

We propose PGGM, a Promptable Geometric Grounding Model for B-Rep models. Given a B-Rep model and a prompt describing modeling intent, PGGM predicts primitive-level grounding logits.

3.1 PROBLEM FORMULATION

Let a B-Rep model be denoted as $B = (\mathcal{F}, \mathcal{E}, \mathcal{C})$, where $\mathcal{F} = \{f_i\}_{i=1}^{N_f}$ is the set of faces, $\mathcal{E} = \{e_j\}_{j=1}^{N_e}$ is the set of edges, and $\mathcal{C}$ denotes the set of oriented coedges that define the local boundary and adjacency structure of the B-Rep. We define the primitive set as $\mathcal{P} = \mathcal{F} \cup \mathcal{E} = \{p_i\}_{i=1}^{N}$, where each primitive p_i is either a face or an edge and $N = N_f + N_e$. The grounding target is a binary label vector $\mathbf{y} \in \{0,1\}^N$, where $y_i = 1$ indicates that primitive p_i is selected.

PGGM supports two prompt signals. A prompt may include a natural language description q, and a partial primitive selection $S^+ \subseteq \mathcal{P}$. We represent the prompt as $p = (q, S^+)$, where the available fields provide complementary cues for primitive-level grounding. Given B and p, the model predicts primitive-wise grounding logits $\mathbf{s} = \mathrm{PGGM}(B, p) \in \mathbb{R}^N$.

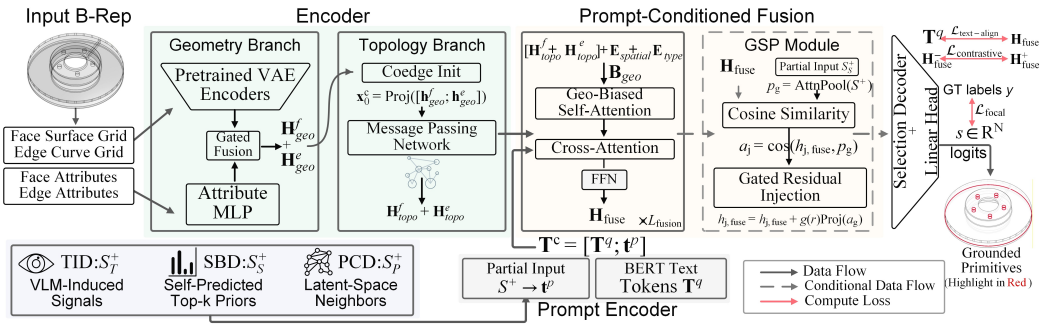


Figure 1: Overview of PGGM for promptable B-Rep primitive grounding. PGGM decouples geometry and topology and supports multi-conditional primitive grounding through prompt-conditioned fusion. During training, discriminative bootstrapping introduces noisy partial signals from VLMs, self-predictions, and latent-space hard negatives, improving robustness against hard-negative ambiguity.

3.2 Promptable Geometric Grounding Model

PGGM consists of four components: a decoupled geometry-topology encoder for continuous geometry and discrete B-Rep connectivity, a prompt encoder for available text and partial-selection cues, a prompt-conditioned fusion decoder for primitive-level prediction, and a Geometric Similarity Propagation (GSP) module that, when partial selections are available, propagates cues from selected primitives to related candidates. The overall framework is illustrated in Fig. 1.

3.2.1 Decoupled Geometry-Topology Encoding

The geometry branch encodes continuous face and edge shape information, while the topology branch performs coedge-based message passing over the discrete B-Rep structure.

Geometry Branch. For each face, the input contains a surface grid $\mathbf{U}^f \in \mathbb{R}^{N_f \times 32 \times 32 \times 3}$ and an explicit face attribute vector $\mathbf{A}^f \in \mathbb{R}^{N_f \times 16}$. For each edge, the input contains a curve grid $\mathbf{U}^e \in \mathbb{R}^{N_e \times 32 \times 3}$ and an explicit edge attribute vector $\mathbf{A}^e \in \mathbb{R}^{N_e \times 13}$. The face attributes include area, face type, centroid, and rationality indicators, while the edge attributes include length, edge type, and convexity indicators.

We use pretrained Variational Autoencoders (VAEs) from Xu et al. (2024) to encode surface and curve grids. The face VAE maps each surface grid to a latent tensor of shape $3 \times 4 \times 4$, which is flattened and projected to the model dimension d. Similarly, the edge VAE maps each curve grid to a latent tensor of shape 3×4, which is flattened and projected to d. The explicit attributes are encoded by separate MLPs. For primitive type $r \in \{f, e\}$, this process can be written as

$$\mathbf{H}^r_{\text{vae}} = \phi^r_{\text{vae}}(\mathbf{U}^r), \qquad \mathbf{H}^r_{\text{attr}} = \phi^r_{\text{attr}}(\mathbf{A}^r). \tag{1}$$

PGGM uses a learned gate to fuse VAE-based geometric and explicit attribute representations:

$$\mathbf{G}^r = \sigma\big(\mathbf{W}^r_g[\mathbf{H}^r_{\text{vae}}; \mathbf{H}^r_{\text{attr}}]\big), \qquad \mathbf{H}^r_{\text{geo}} = \mathbf{G}^r \odot \mathbf{H}^r_{\text{vae}} + (1 - \mathbf{G}^r) \odot \mathbf{H}^r_{\text{attr}}. \tag{2}$$

This gate allows the model to adaptively balance dense local shape information from the pretrained VAE and compact handcrafted geometric attributes for each primitive. The resulting geometry embeddings are denoted as $\mathbf{H}^f_{\text{geo}} \in \mathbb{R}^{N_f \times d}$ and $\mathbf{H}^e_{\text{geo}} \in \mathbb{R}^{N_e \times d}$.

Topology Branch. The topology branch performs coedge-based message passing Lambourne et al. (2021) to capture structured relations among faces, edges, and coedges. For each coedge $c \in \mathcal{C}$, PGGM initializes its feature from the geometry embeddings of its incident face and edge:

$$\mathbf{x}^c_0 = \text{Proj}_t\left([\mathbf{H}^f_{\text{geo}}[\text{face}(c)]; \mathbf{H}^e_{\text{geo}}[\text{edge}(c)]]\right). \tag{3}$$

The initial face and edge states are $\mathbf{X}^f_0 = \mathbf{H}^f_{\text{geo}}$ and $\mathbf{X}^e_0 = \mathbf{H}^e_{\text{geo}}$. At each topology layer, for each coedge c, features from adjacent faces, edges, and coedges are gathered via predefined B-Rep graph walks, with neighbor sets $\mathcal{N}^f(c)$, $\mathcal{N}^e(c)$, and $\mathcal{N}^c(c)$. These features are concatenated and passed through a shared MLP to produce update vectors for coedges, edges, and faces:

$$[\mathbf{z}^{\text{co}}_c; \mathbf{z}^e_c; \mathbf{z}^f_c] = \text{MLP}_{\text{topo}}\left([\mathbf{X}^f_\ell[\mathcal{N}^f(c)]; \mathbf{X}^e_\ell[\mathcal{N}^e(c)]; \mathbf{X}^c_\ell[\mathcal{N}^c(c)]]\right). \tag{4}$$

The coedge state is updated directly, while edge and face states are updated by max-pooling over their associated coedges. After L_{t} layers, the topology-enriched primitive embeddings are $\mathbf{H}^{f}_{\mathrm{topo}} = \mathbf{X}^{f}_{L_{\mathrm{t}}}$ and $\mathbf{H}^{e}_{\mathrm{topo}} = \mathbf{X}^{e}_{L_{\mathrm{t}}}$. These embeddings preserve local geometric evidence while incorporating structured primitive relations, and are used as primitive tokens for prompt-conditioned grounding.

3.2.2 Prompt Encoding

PGGM encodes natural-language descriptions and partial primitive selections as unified condition tokens, which serve as the key-value sequence in prompt-conditioned cross-attention. The two prompt modalities can be used independently or jointly.

Text Prompt. When a natural language description q is available, PGGM encodes it with a pre-trained BERT encoder Devlin et al. (2019). To capture fine-grained correspondences between words and primitives, PGGM keeps all text tokens and projects them to the model dimension, $\mathbf{T}^{q} = \mathrm{Proj}_{q}(\mathrm{BERT}(q)) \in \mathbb{R}^{L_q \times d}$.

Partial Selection Prompt. When a partial selection S^{+} is provided, PGGM converts the selected primitives into a prototype token. The selected face and edge embeddings are gathered from the topology-branch outputs $\mathbf{H}^{f}_{\mathrm{topo}}$ and $\mathbf{H}^{e}_{\mathrm{topo}}$. Let $\mathbf{h}_i$ denote the topology embedding of the i-th selected primitive. The partial prototype is computed by attention pooling:

$$\alpha_i = \mathrm{softmax}_i\left(\mathbf{v}_p^{\top}\tanh(\mathbf{W}_p\mathbf{h}_i)\right), \qquad \mathbf{t}^{p} = \mathrm{Proj}_{p}\big(\textstyle\sum_{i\in S^{+}} \alpha_i \mathbf{h}_i\big) + \mathbf{e}_p \tag{5}$$

where $\mathbf{e}_p$ is a learnable type embedding indicating that the token comes from a positive partial selection. During the discriminative bootstrapping stage introduced in Sec. 3.3, PGGM uses an additional reliability token $\mathbf{t}^{r}$ to indicate the trustworthiness of partial cues derived from noisy auxiliary signals. The final condition sequence is the concatenation of all prompt tokens:

$$\mathbf{T}^{c} = [\mathbf{T}^{q}; \mathbf{t}^{p}; \mathbf{t}^{r}], \qquad \mathbf{T}^{c} \in \mathbb{R}^{L_c \times d}. \tag{6}$$

3.2.3 Prompt-Conditioned Fusion and Decoding

After geometry-topology encoding, PGGM concatenates face and edge topology embeddings into a unified primitive sequence, $\mathbf{H}^{0} = [\mathbf{H}^{f}_{\mathrm{topo}}; \mathbf{H}^{e}_{\mathrm{topo}}] + \mathbf{E}_{\mathrm{type}} + \mathbf{E}_{\mathrm{spatial}}$, where $\mathbf{E}_{\mathrm{type}}$ is a learnable primitive-type embedding that distinguishes faces from edges, and $\mathbf{E}_{\mathrm{spatial}}$ is an oriented spatial positional encoding computed from primitive-level spatial attributes. Specifically, for faces, we use centroids and normals; for edges, we use midpoints and curve directions.

The fusion module updates primitive tokens through stacked transformer-style layers. Each layer first performs geometry-biased self-attention over primitives, then performs cross-attention from primitive tokens to condition tokens, followed by a feed-forward network:

$$\mathbf{H}^{\ell+1} = \mathrm{FFN}\left(\mathrm{CA}\left(\mathrm{GSA}\left(\mathbf{H}^{\ell}; \mathbf{B}_{\mathrm{geo}}\right), \mathbf{T}^{c}\right)\right). \tag{7}$$

Here, GSA denotes geometry-biased self-attention, CA denotes prompt cross-attention, and FFN denotes a feed-forward network. All layers are implemented with pre-norm and residual connections.

We use a geometry bias $\mathbf{B}_{\mathrm{geo}}$ as a geometry-aware prior for primitive self-attention, constructed from geometry-branch embeddings: face-face and edge-edge entries use cosine similarities of $\mathbf{H}^{f}_{\mathrm{geo}}$ and $\mathbf{H}^{e}_{\mathrm{geo}}$, while face-edge entries are zero:

$$\mathbf{B}_{\mathrm{geo}}(i,j) = \mathbf{1}[\tau(p_i) = \tau(p_j)]\cos(\mathbf{h}_{i,\mathrm{geo}}, \mathbf{h}_{j,\mathrm{geo}}), \quad \tau(p) \in \{f, e\}. \tag{8}$$

After L_{fusion} fusion layers, PGGM obtains prompt-conditioned primitive embeddings $\mathbf{H}_{\mathrm{fuse}}$. A selection decoder further refines these embeddings for grounding, and a linear selection head $\mathrm{Head}_{\mathrm{sel}}$ maps each decoded primitive token to a scalar logit:

$$\mathbf{s} = \mathrm{Head}_{\mathrm{sel}}\left(\mathrm{Decoder}\left(\mathbf{H}_{\mathrm{fuse}}, \mathbf{T}^{c}\right)\right), \qquad \mathbf{s} \in \mathbb{R}^{N_f + N_e}. \tag{9}$$

3.2.4 Geometric Similarity Propagation

Partial selections provide direct examples of target primitives. To exploit this signal, PGGM introduces Geometric Similarity Propagation (GSP), applied after prompt-conditioned fusion and before decoding. GSP is used only when a partial prompt is available.

Given fused primitive embeddings $\mathbf{H}_{\text{fuse}}$ and a partial selection S^+, GSP first aggregates selected primitive embeddings into a prototype by attention pooling, $\mathbf{p}_g = \text{AttnPool}\left(\{\mathbf{h}_{i,\text{fuse}}\}_{i\in S^+}\right)$. GSP then scores every primitive in the same B-Rep by its cosine similarity to the prototype, $a_j = \cos(\mathbf{h}_{j,\text{fuse}}, \mathbf{p}_g)/\tau$, where τ is a learnable temperature initialized to a small value. The scalar similarity score is projected to the model dimension and injected into the primitive embedding by residual addition, $\widetilde{\mathbf{h}}_j = \mathbf{h}_{j,\text{fuse}} + \text{Proj}_s(a_j)$. When a global reliability score r is provided by the DB training signals described in Sec. 3.3, PGGM gates the propagation strength:

$$\mathbf{h}_j^{\text{gsp}} = \mathbf{h}_{j,\text{fuse}} + g(r)\big(\widetilde{\mathbf{h}}_j - \mathbf{h}_{j,\text{fuse}}\big), \quad g(r) = \sigma(\mathbf{w}_r r + b_r). \tag{10}$$

The resulting embeddings $\mathbf{H}_{\text{gsp}}$ are used by the selection decoder for final primitive-level grounding.

3.3 Discriminative Bootstrapping

To address hard-negative ambiguity, we introduce Discriminative Bootstrapping (DB), which trains text-conditioned samples with noisy auxiliary signals bootstrapped from the same textual input.

3.3.1 Noisy Partial Signal Construction

DB constructs partial prompts from multiple sources, each providing a set of selected primitives, associated confidence scores, and a global reliability estimate, forming a complementary supervision space that systematically exposes the model to diverse sources of hard-negative ambiguity.

Teacher-Induced Discrimination (TID). Vision-Language Models (VLMs) provide a source of semantic and visual ambiguity by selecting primitives that are plausible with respect to the text description and rendered B-Rep context. DB exploits this to construct teacher-induced hard partial prompts: given the model's text-conditioned prediction, we take the top-K candidate primitives, render them with the B-Rep context, and ask the VLM to identify positive candidates with discrete confidence levels. These predictions form the teacher partial selection $S_{\text{T}}^+ = S_{\text{T}}^f \cup S_{\text{T}}^e$. Each discrete confidence is mapped to a scalar $c_i \in [0, 1]$, and the global reliability is computed as $r_{\text{T}} = |S_{\text{T}}^+|^{-1} \sum_{i\in S_{\text{T}}^+} c_i$. Since VLM-selected primitives can be visually or semantically plausible without matching the ground truth, this source exposes the model to teacher-induced hard negatives.

Self-Bootstrapped Discrimination (SBD). Given a text-conditioned prediction, the model selects the top-k primitives according to its predicted grounding probabilities. These primitives form a self-bootstrapped partial prompt: $S_{\text{S}}^+ = \text{TopK}(\sigma(\mathbf{s}_{\text{text}}), k), \quad r_{\text{S}} = \frac{1}{k}\sum_{i\in S_{\text{S}}^+} \sigma(s_i)$. The per-primitive confidence is defined as $c_i = \sigma(s_i)$ for each $i \in S_{\text{S}}^+$, and the global reliability is the mean confidence of the selected primitives. During training, SBD is periodically refreshed from the current model. This source exposes the model to its own failure modes, encouraging it to become robust to model-induced ambiguity rather than reinforcing early mistakes blindly.

Proximity-Contrastive Discrimination (PCD). PCD constructs noisy partial prompts by combining a random subset of ground-truth positives with hard negatives that are geometrically close to the target. Let $\mathcal{P}^+ = \{p \in \mathcal{P} \mid y_p = 1\}$ denote the ground-truth positive primitive set. DB first computes a positive prototype from all ground-truth positives, $\mathbf{p}_+ = |\mathcal{P}^+|^{-1} \sum_{p\in\mathcal{P}^+} \mathbf{h}_p$, and retrieves hard negatives as the top-k non-target primitives with the highest cosine similarity to this prototype:

$$S_{\text{hard}}^- = \text{TopK}\big(\{\cos(\mathbf{h}_p, \mathbf{p}_+)\}_{p\in\mathcal{P}\setminus\mathcal{P}^+}, k\big). \tag{11}$$

For each noisy partial prompt, PCD samples a positive subset $S_{\text{gt}}^+ \subseteq \mathcal{P}^+$ of size about $K/2$ and fills the remaining slots with hard negatives, yielding $S_{\text{P}}^+ = S_{\text{gt}}^+ \cup S_{\text{hard}}^-$ with approximately K primitives. The per-primitive confidence is set to $c_i = 1$ for sampled positives and $c_i = \text{clamp}((\text{sim}_i+1)/2, 0, 1)$ for hard negatives, where $\text{sim}_i = \cos(\mathbf{h}_i, \mathbf{p}_+)$. The global reliability is defined as the average confidence of the prompt, $r_{\text{P}} = |S_{\text{P}}^+|^{-1} \sum_{i\in S_{\text{P}}^+} c_i$. This source encourages the model to separate true target primitives from geometrically similar distractors.

Training. For each source $X \in \{\text{T}, \text{S}, \text{P}\}$, the constructed signal is represented as a partial prompt

$$\mathcal{A}_X = (S_X^+, \{c_i\}_{i\in S_X^+}, r_X). \tag{12}$$

During training, $\mathcal{A}_X$ is combined with the textual description q and fed into PGGM as a text-and-partial condition, while the model is supervised by the ground-truth label vector $\mathbf{y}$.

3.3.2 RELIABILITY-AWARE TRAINING

DB makes the reliability of partial prompts explicit through primitive-level confidence and global reliability. For a selected primitive embedding $\mathbf{h}_i$ with confidence c_i, the partial-selection encoder uses $\hat{\mathbf{h}}_i = \mathbf{h}_i + \mathrm{Proj}_c(c_i)$ before attention pooling, and adds c_i to the pooling score so reliable selected primitives contribute more strongly to the partial prototype. The global reliability score r is encoded as an additional reliability token, $\mathbf{t}^r = \mathrm{Proj}_r(r) + \mathbf{e}_r$, and also gates geometric similarity propagation in Eq. 10, reducing the influence of unreliable partial prompts.

DB further adds a consistency objective for reliable noisy partial variants. Let $\mathbf{s}_{\mathrm{clean}}$ be the logits produced by a clean partial prompt, and let $\mathbf{s}_{\mathrm{noisy}}$ be the logits produced by a noisy partial prompt. When the noisy prompt has global reliability above a threshold η, DB encourages its prediction to stay close to the detached clean reference:

$$\mathcal{L}_{\mathrm{cons}} = \lambda_{\mathrm{cons}}\, r\, \mathrm{KL}\left(\sigma(\mathbf{s}_{\mathrm{clean}}) \parallel \sigma(\mathbf{s}_{\mathrm{noisy}})\right), \qquad r \geq \eta. \tag{13}$$

3.4 TRAINING OBJECTIVE

PGGM is trained with primitive-level supervision over the ground-truth label $\mathbf{y}$. The main grounding objective is a focal loss over primitive logits. For each primitive, with binary cross entropy $\mathrm{BCE}(s_i, y_i)$ and $p_i^t = \exp(-\mathrm{BCE}(s_i, y_i))$, the focal term is $\mathcal{L}_{\mathrm{focal}} = \frac{1}{N}\sum_i \alpha_i (1-p_i^t)^\gamma \mathrm{BCE}(s_i, y_i)$, where $\alpha_i = \alpha y_i + (1-\alpha)(1-y_i)$.

PGGM also uses a margin-based contrastive loss in the prompt-conditioned primitive embedding space. Given positive and negative primitive embeddings, let $\mathbf{p}_+$ be the normalized mean positive prototype, and let s_+ and s_- denote the mean cosine similarities of positive and negative primitives to $\mathbf{p}_+$. The contrastive loss is $\mathcal{L}_{\mathrm{con}} = \max(0, s_- - s_+ + m)$. For text-conditioned variants, PGGM aligns language and geometry using a symmetric text-primitive InfoNCE loss. For each valid sample, text tokens are pooled into a text embedding $\mathbf{t}_b$, and GT-positive primitive embeddings are pooled into a positive primitive prototype $\mathbf{p}_b$. With normalized embeddings and temperature τ_{text}, the similarity matrix is $z_{bc} = \mathbf{t}_b^\top \mathbf{p}_c / \tau_{\mathrm{text}}$, and the text alignment loss is $\mathcal{L}_{\mathrm{text}} = \frac{1}{2}[\mathrm{CE}(z, I) + \mathrm{CE}(z^\top, I)]$.

The full training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{focal}} + \lambda_{\mathrm{con}}\mathcal{L}_{\mathrm{con}} + \lambda_{\mathrm{text}}\mathcal{L}_{\mathrm{text}} + \mathcal{L}_{\mathrm{cons}}. \tag{14}$$

Losses are averaged over all condition variants, which are constructed by randomly dropping the text or sampled partial component.

4 EXPERIMENTS

We evaluate PGGM on B-Rep grounding under diverse conditions and analyze the contributions of its main components. We further integrate PGGM into the backend of the text-to-CAD pipeline FutureCAD Li et al. (2026) and observe improved downstream generation quality, as reported in Appendix B.

4.1 EXPERIMENTAL SETUP

Datasets. We conduct experiments on FutureCAD Li et al. (2026), a benchmark derived from real CAD workflows that captures user selections during interactive modeling. To increase data diversity, we build a large-scale pretraining corpus from ABC Koch et al. (2019) using FilletRec Gao et al. (2025), where local features (e.g. *fillet* and *chamfer*) are detected and removed, and the resulting sharp edges are treated as ground-truth primitives and paired with automatically generated textual descriptions. All methods, including baselines, are pretrained on this corpus. Details of the dataset are in Appendix A.

Implementation Details. PGGM employs a 512-dimensional primitive representation. The topology branch comprises six coedge message-passing layers, eight prompt-conditioned fusion layers, and a six-layer selection decoder, all using eight attention heads. Pretrained VAEs Xu et al. (2024) and BERT Devlin et al. (2019) are kept frozen and adapted via low-rank adaptation Hu et al. (2021). Training uses AdamW with an initial learning rate of 10^{-4}. For partial conditioning, we randomly

Table 1: Results on text-conditioned B-Rep grounding. We report Recall@K (R@K), Precision (P), mean Average Precision (mAP), and F1 score. All metrics are reported in percentage (%).

Methods	R@3↑	R@5↑	R@10↑	P↑	mAP↑	F1↑
EDA Wu et al. (2023)	30.40	38.30	48.70	15.45	50.33	23.45
BRepMFR Zhang et al. (2024)	55.68	67.96	79.86	44.83	62.90	51.75
BRepFormer Dai et al. (2025)	56.59	68.29	79.96	46.69	63.68	51.59
BRepGround Li et al. (2026)	58.20	67.03	80.29	41.95	64.53	51.12
VLM-Verifier	30.88	43.42	60.32	10.82	35.32	19.11
PGGM (Ours)	**68.45**	**78.46**	**87.57**	**70.42**	**76.97**	**69.01**

Table 2: Multi-condition B-Rep grounding results. PGGM is evaluated with text, partial, and text+partial prompts to assess prompt complementarity and the contribution of each component.

Methods	**Text**				**Partial**				**Text+Partial**			
	R@5↑	P↑	mAP↑	F1↑	R@5↑	P↑	mAP↑	F1↑	R@5↑	P↑	mAP↑	F1↑
PGGM (full)	**78.46**	**70.42**	**76.97**	**69.01**	**91.84**	83.17	94.49	79.11	90.77	**87.44**	**94.48**	**84.90**
w/o DB	71.41	65.56	69.52	60.56	91.27	**84.59**	**95.83**	81.75	90.15	83.73	93.34	82.81
w/o GSP	75.83	62.80	70.53	65.36	91.19	84.14	93.54	**81.89**	91.50	80.30	94.13	84.42
w/o $\mathcal{L}_{\text{text}}$	69.71	52.47	65.85	55.63	91.41	78.16	92.44	81.64	**91.97**	82.25	94.06	82.74
w/o $\mathcal{L}_{\text{con}}$	74.52	57.58	69.38	60.02	91.78	76.45	93.73	78.58	91.37	80.18	91.80	81.93

sample 10%-40% of ground-truth primitives for each sample. We warm up with clean partials and then continue with DB training, using TID signals collected once with Qwen3-VL-32B Bai et al. (2025) after warmup, SBD refreshed every epoch, and PCD refreshed every two epochs. Additional hyperparameters are provided in Appendix D.

Baselines. We compare with three categories of methods. (i) EDA Wu et al. (2023) is adapted from 3D visual grounding by using point clouds sampled from B-Rep models as input and treating the axis-aligned bounding boxes of selected B-Rep primitives as grounding targets. (ii) Feature recognition methods, including BRepMFR Zhang et al. (2024) and BRepFormer Dai et al. (2025), are extended with a BERT Devlin et al. (2019) text encoder and a primitive-level grounding decoder. (iii) BRepGround Li et al. (2026) serves as the most directly related baseline. We further include a VLM-based verifier baseline built on Gemini-2.5-Flash Google (2026) for verifying primitives. Additional adaptation details are provided in Appendix C.

4.2 Main Results under Text-Conditioned Grounding

Table 1 compares PGGM with representative baselines under text-conditioned grounding. PGGM consistently achieves the best performance across all metrics, with improvements of 17.26 in F1 and 23.73 in precision over the strongest baseline. The gains suggest that PGGM induces a more discriminative geometry-topology representation space, allowing semantically relevant primitives to be separated despite discrete topology, local adjacency, and geometric similarity.

EDA performs poorly, achieving only 23.45 F1, indicating that existing 3D visual grounding methods based on point-cloud object localization fail to capture the primitive identities and topological structure required by B-Rep grounding, and thus require task-specific architectural adaptation. BRepMFR and BRepFormer achieve relatively strong performance due to their native B-Rep inputs, but their results remain suboptimal, suggesting that existing machining feature recognition models can extract useful local geometric features yet struggle with text-geometry alignment and hard-negative discrimination, as reflected by their lower precision. PGGM also outperforms BRepGround, which suffers from similar issues in alignment and discrimination. Despite the strong scene understanding and reasoning capabilities of VLMs, VLM-Verifier struggles in B-Rep grounding, as the task requires distinguishing many interwoven primitives with subtle geometric differences, and exhaustive VLM-based candidate verification is costly at primitive scale. These results suggest that PGGM learns more structured primitive representations through decoupled geometry-topology encoding, which are further refined by discriminative bootstrapping under hard-negative ambiguity.

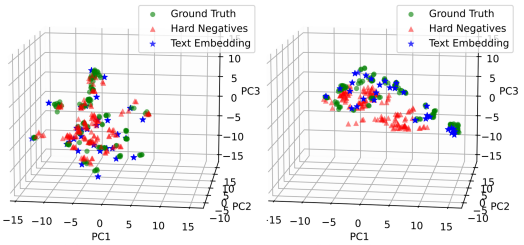
4.3 Multi-Condition Grounding

Table 2 evaluates PGGM under text, partial, and text+partial conditions. Partial-only grounding reaches 79.11 F1, confirming that a sparse primitive selection provides a strong geometric reference

Table 3: Discriminative Bootstrapping ablation on text-conditioned B-Rep grounding. Teacher-guided variants use Gemini-2.5-Flash or Qwen3-VL-32B to verify the top-5 text-only predictions and construct partial prompts from the verified positives.

Methods	R@5↑	P↑	mAP↑	F1↑
PGGM (full)	**78.46**	**70.42**	**76.97**	**69.01**
+Gemini	79.66 (+1.20)	73.15 (+2.73)	78.08 (+1.11)	70.06 (+1.05)
+Qwen	80.23 (+1.77)	73.89 (+3.47)	77.47 (+0.50)	69.87 (+0.86)
w/o DB	71.41	65.56	69.52	60.56
w/o TID	77.52	68.49	75.02	67.96
w/o SBD	74.35	62.06	70.28	64.43
w/o PCD	76.24	63.41	72.35	65.33

Figure 2: Visualization of primitive and text embeddings before and after DB training, respectively. The DB-trained model learns a more discriminative representation distribution.



(a) w/o DB (b) with DB

for inferring the remaining targets. Combining text with partial prompts further improves F1 to 84.90, showing that the two forms of supervision capture complementary information.

The ablation results further isolate the contribution of each component. Removing DB leads to a significant decrease in text-only F1 and a notable drop in precision, demonstrating that DB directly strengthens discrimination when grounding relies on text. The w/o DB variant attains higher scores in some clean partial-only metrics, including F1. This suggests that DB does not simply optimize for clean partial-prompt fitting; rather, it trades a small amount of clean partial performance for stronger robustness in text-conditioned and text+partial grounding. Removing GSP lowers text-only F1 to 65.36, indicating that partial prototype propagation also contributes to representation learning during training. At the same time, the partial-only gains of w/o GSP suggest that similarity propagation can occasionally amplify nearby negative primitives. Nevertheless, the full model performs better overall, particularly in settings where text participates. Finally, removing $\mathcal{L}_{\text{text}}$ leads to a substantial decrease in text-only F1, dropping to 55.63, underscoring the importance of language-geometry alignment. Removing $\mathcal{L}_{\text{con}}$ also results in a noticeable decline in text-only F1 to 60.02 and a reduction in precision, indicating that explicit contrastive separation is important for distinguishing hard negatives.

4.4 Discriminative Bootstrapping Analysis

Table 3 provides a closer analysis of DB. We first study teacher-guided inference. Given the text-only prediction, we select the top-5 candidate primitives and ask VLMs to identify positives among them. The verified positives are then combined with the original text for text+partial inference. Using either Gemini-2.5-Flash Google (2026) or Qwen3-VL-32B Bai et al. (2025) as the teacher brings consistent gains over text-only PGGM, with noticeable precision improvements. This suggests that DB makes PGGM robust to noisy auxiliary cues and allows it to extract useful guidance from VLM feedback.

The component ablations suggest that each source in DB provides complementary supervision. Removing any individual component leads to consistent degradation, indicating that effective learning requires jointly modeling multiple ambiguity regimes. TID introduces externally induced semantic ambiguity, SBD reflects the model's own error distribution, and PCD constrains the representation space by injecting hard negatives near the target manifold. Qualitative visualization in Fig. 2 further suggests that DB encourages a more discriminative embedding distribution. These sources act as complementary constraints on the decision boundary, collectively shaping a representation space that remains discriminative under fine-grained geometric and semantic ambiguity.

5 Conclusion

We introduced **PGGM**, a promptable geometric grounding model for primitive-level B-Rep grounding. PGGM jointly models continuous geometry and discrete topology through decoupled encoding, prompt-conditioned grounding, and geometric similarity propagation for partial selections. To improve fine-grained discrimination, we proposed Discriminative Bootstrapping, a general training strategy that turns bootstrapped conditional signals from external, self-generated, and representation-near sources into discriminative supervision for stronger grounding and hard-negative separation. Experiments show that PGGM consistently outperforms adapted 3D grounding methods, B-Rep feature recognition models, prior B-Rep grounding methods, and VLM-based verification.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and as an auxiliary tool for code development. Generative AI was not used to write the scientific content of the paper, formulate the research problem, develop the proposed methodology, or design the core architecture of the implementation. These aspects of the work were carried out by the authors. Generative AI models were also used in our experiments for auxiliary training signal construction, teacher-guided inference, and VLM-based baseline evaluation. We take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Our model architecture and training objectives are described in Section 3. Dataset construction and preprocessing are detailed in Appendix A. Baseline adaptations are documented in Appendix C. Model and optimization hyperparameters are provided in Appendix D. Computational resources for model training and teacher-signal construction are reported in Appendix E.

REFERENCES

Panos Achlioptas, Ahmed Abdelreheem, F. Xia, Mohamed Elhoseiny, and Leonidas J. Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European Conference on Computer Vision*, 2020.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Dave Zhenyu Chen, Angel X. Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. *ArXiv*, abs/1912.08830, 2019.

John G Cherng, Xin-Yu Shao, Yubao Chen, and Peter R Sferro. Feature-based part modeling and process planning for rapid response manufacturing. *Computers & industrial engineering*, 34(2): 515–530, 1998.

Andrew R. Colligan, Trevor T. Robinson, Declan C. Nolan, Yang Hua, and Weijuan Cao. Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Comput. Aided Des.*, 147:103226, 2022.

Amirmohammad Daareyni, Antti Martikkala, Hossein Mokhtarian, and Iñigo Flores Ituarte. Generative ai meets cad: enhancing engineering design to manufacturing processes with large language models. *The International Journal of Advanced Manufacturing Technology*, pp. 1–10, 2025.

Yongkang Dai, Xiaoshui Huang, Yunpeng Bai, Hao Guo, Hongping Gan, Ling Yang, and Yilei Shi. Brepformer: Transformer-based b-rep geometric feature recognition. *Proceedings of the 2025 International Conference on Multimedia Retrieval*, 2025.

Liyuan Deng, Hao Guo, Yunpeng Bai, Yongkang Dai, Huaxi Huang, and Yilei Shi. Brepllm: Native boundary representation understanding with large language models, 2025. URL https://arxiv.org/abs/2512.16413.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Jialin Gao, Taoran Liu, Hongfei Ye, and Jianjun Chen. Filletrec: A lightweight graph neural network with intrinsic features for automated fillet recognition. *ArXiv*, abs/2511.05561, 2025.

Google. Gemini 2.5 flash. https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash, 2026. Accessed: 2026-04-26.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

Hsiang-Wei Huang, Kuang-Ming Chen, Wenhao Chai, Cheng-Yen Yang, Jen-Hao Cheng, and Jenq-Neng Hwang. Reasoning matters for 3d visual grounding. *ArXiv*, abs/2601.08811, 2026.

Ronggang Huang, Haoxin Yang, Yan Cai, Xuemiao Xu, Huaidong Zhang, and Shengfeng He. Viewsrd: 3d visual grounding via structured multi-view decomposition. *ArXiv*, abs/2507.11261, 2025.

Shijia Huang, Yilun Chen, Jiaya Jia, and Liwei Wang. Multi-view transformer for 3d visual grounding. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15503–15512, 2022.

Pradeep Kumar Jayaraman, Aditya Sanghi, J. Lambourne, T. Davies, and Hooman Shayani. Uv-net: Learning from boundary representations. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11698–11707, 2021.

Pradeep Kumar Jayaraman, Joseph G. Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. Solidgen: An autoregressive model for direct b-rep synthesis, 2023. URL https://arxiv.org/abs/2203.13944.

Benjamin T. Jones, Michael Hu, Vladimir G. Kim, and Adriana Schulz. Self-supervised representation learning for cad. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21327–21336, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003, 2023.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9593–9603, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Joseph G Lambourne, Karl DD Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12773–12782, 2021.

Jiahao Li, Qingwang Zhang, Qiuyu Chen, Guozhan Qiu, Yunzhong Lou, and Xiangdong Zhou. Towards high-fidelity cad generation via llm-driven program generation and text-based b-rep primitive grounding. *arXiv preprint arXiv:2603.11831*, 2026.

Jing Li, Yihang Fu, and Falai Chen. Dtgbrepgen: A novel b-rep generative model through decoupling topology and geometry. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21438–21447, 2025a.

Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Zero-shot 3d visual grounding from vision-language models. *ArXiv*, abs/2505.22429, 2025b.

Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Seeground: See and ground for zero-shot open-vocabulary 3d visual grounding. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3707–3717, 2025c.

Yilin Liu, Duoteng Xu, Xingyao Yu, Xiang Xu, Daniel Cohen-Or, Hao Zhang, and Hui Huang. Hola: B-rep generation using a holistic latent representation. *ACM Transactions on Graphics (TOG)*, 44: 1 – 25, 2025.

Dacheng Qi, Chenyu Wang, Jingwei Xu, Tianzhe Chu, Zibo Zhao, Wen Liu, Wenrui Ding, Yi Ma, and Shenghua Gao. Pointer-cad: Unifying b-rep and command sequences via pointer-based edges & faces selection. *arXiv preprint arXiv:2603.04337*, 2026.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Tianxu Wang, Zhuofan Zhang, Ziyu Zhu, Yue Fan, Jing Xiong, Pengxiang Li, Xiaojian Ma, and Qing Li. From objects to anywhere: A holistic benchmark for multi-level visual grounding in 3d scenes. *ArXiv*, abs/2506.04897, 2025.

Fei wei Qin, Gaoyang Zhan, Meie Fang, C. L. Philip Chen, and Ping Li. Vgnet: Multimodal feature extraction and fusion network for 3d cad model retrieval. *IEEE Transactions on Multimedia*, 27: 1432–1447, 2025.

Yanmin Wu, Xinhua Cheng, Renrui Zhang, Zesen Cheng, and Jian Zhang. Eda: Explicit text-decoupling and dense alignment for 3d visual grounding. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19231–19242, 2023.

Xiang Xu, Joseph G. Lambourne, Pradeep Kumar Jayaraman, Zhengqing Wang, Karl D. D. Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry, 2024. URL https://arxiv.org/abs/2401.15563.

Xiang Xu, Pradeep Kumar Jayaraman, J. Lambourne, Yilin Liu, Durvesh Malpure, and Pete Meltzer. Autobrep : Autoregressive b-rep generation with unified topology and geometry. *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, 2025.

Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F. Fouhey, and Joyce Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7694–7701, 2024.

Canrong Yao, Kang Wu, Zuheng Zheng, Siyuan Xing, and Xiao-Ming Fu. Brepmae: Self-supervised masked brep autoencoders for machining feature recognition. *ArXiv*, abs/2602.22701, 2026.

Tatiana Zemskova and Dmitry A. Yudin. 3dgraphllm: Combining semantic graphs and large language models for 3d scene understanding. *ArXiv*, abs/2412.18450, 2024.

Shuming Zhang, Zhidong Guan, Hao Jiang, Xiaodong Wang, and Pingan Tan. Brepmfr: Enhancing machining feature recognition in b-rep models through deep learning and domain adaptation. *Comput. Aided Geom. Des.*, 111:102318, 2024.

Lichen Zhao, Daigang Cai, Lu Sheng, and Dong Xu. 3dvg-transformer: Relation modeling for visual grounding on point clouds. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2908–2917, 2021.

A Dataset Details

We use the grounding dataset provided by FutureCAD Li et al. (2026). The dataset contains 79,641 training samples, 4,424 validation samples, and 4,424 test samples. Each grounding sample consists of a B-Rep model, a natural-language description, and the corresponding primitive-level labels. The source modeling operations include *fillet*, *chamfer*, and *shell*.

As discussed in the main paper, we further use the ABC dataset to improve data diversity. Specifically, we follow MFCAD++ Colligan et al. (2022) to generate CAD models containing *chamfer* and *fillet* features. We then apply and extend the detection method of FilletRec Gao et al. (2025) to identify these local features, and use OpenCASCADE to defeature the models. The newly created sharp edges after defeaturing are treated as primitive-level labels. This process yields 235,598 additional training

samples, including 161,793 *fillet* samples and 73,805 *chamfer* samples. Since FutureCAD is also derived from the ABC dataset, we remove any samples whose IDs overlap with those in FutureCAD to prevent potential data leakage.

Table 4: Effect of ABC-derived pretraining data on multi-condition B-Rep grounding. Additional pretraining data improves grounding performance by increasing geometric diversity and exposing models to more primitive-level feature patterns.

Methods	**Text**				**Partial**				**Text+Partial**			
	R@5↑	P↑	mAP↑	F1↑	R@5↑	P↑	mAP↑	F1↑	R@5↑	P↑	mAP↑	F1↑
PGGM (full)	78.46	70.42	76.97	69.01	91.84	83.17	94.49	79.11	90.77	87.44	94.48	84.90
w/o ABC	78.38	69.57	76.19	68.28	90.29	82.11	93.41	78.76	90.03	86.56	94.10	84.36

We use the ABC-derived data only for pretraining all methods, including the baselines, and then train them on the FutureCAD training set. The validation and test sets contain only FutureCAD samples. Table 4 reports an ablation on the additional ABC-derived data. The performance improvement after using this data indicates that large-scale synthetic pretraining improves the robustness and generalization of B-Rep grounding models, especially by exposing them to more diverse geometric configurations and primitive-level feature patterns. The ablation further indicates that ABC-derived pretraining is beneficial but is not the primary driver of the performance gains, highlighting the contribution of the proposed architecture and training strategy.

Table 5: Downstream text-to-CAD generation results after replacing BRepGround with PGGM in the FutureCAD Li et al. (2026) pipeline. PGGM improves operation-level F1, reduces Chamfer Distance (CD), and lowers the invalid ratio (IR), demonstrating its effectiveness as a backend grounding module for text-to-CAD generation.

Methods	*G*-F1↑	*R*-F1↑	*Median* CD↓	*Mean* CD↓	IR(%)↓
FutureCAD + BRepGround	80.54	85.35	31.12	71.39	1.01
FutureCAD + PGGM	**81.80**	**88.41**	**27.32**	**64.55**	**0.00**

B Integration with Text-to-CAD Generation

To evaluate whether PGGM can improve downstream text-to-CAD generation, we integrate it into the backend of FutureCAD Li et al. (2026). Specifically, we replace the original BRepGround module in FutureCAD with PGGM and retrain the pipeline under the same setting. The results are reported in Table 5.

Replacing BRepGround with PGGM improves both grounding-related and reconstruction-related metrics. FutureCAD + PGGM achieves higher *G*-F1 and *R*-F1, indicating more accurate operation grounding and reconstruction quality. It also reduces both median and mean Chamfer Distance, suggesting that more accurate primitive grounding leads to generated CAD models that better match the target geometry. In addition, PGGM reduces the invalid ratio from 1.01% to 0.00%, showing that improved primitive-level grounding can also benefit the structural validity of generated CAD models. These results demonstrate that PGGM is not only effective as a standalone grounding model, but also improves the quality of a practical text-to-CAD generation pipeline.

C Baseline Implementations

C.1 EDA

EDA Wu et al. (2023) is originally designed for language-guided 3D object grounding in indoor scenes. We adapt it to our B-Rep grounding task with minimal architectural changes. For each CAD model, we uniformly sample 10,240 points from the B-Rep geometry to form a point cloud, which is then used as the 3D input to EDA. The textual input is the natural-language description associated

with the model. Only selected primitives are used to construct ground-truth boxes for training, while both selected and unselected primitives are considered as candidates during primitive-level evaluation.

Specifically, given a sampled point cloud X and a textual query q, EDA predicts a fixed set of box proposals $\{\hat{b}_k\}_{k=1}^{K}$, where each box is parameterized by its center and size:

$$\hat{b}_k = (\hat{c}_k, \hat{r}_k), \quad \hat{c}_k \in \mathbb{R}^3,\ \hat{r}_k \in \mathbb{R}^3. \tag{15}$$

The ground-truth box set is built from the selected B-Rep primitives:

$$\mathcal{B}^+ = \{b_i = \mathrm{AABB}(p_i) \mid y_i = 1\}, \tag{16}$$

where p_i denotes a face or edge primitive and $y_i \in \{0, 1\}$ is its binary selection label.

During training, predicted boxes are matched to ground-truth primitive boxes using Hungarian matching, following the original DETR-style formulation in EDA. The box regression objective consists of an ℓ_1 loss and a generalized IoU loss:

$$\mathcal{L}_{\text{box}} = \sum_i \left(\lambda_1 \|\hat{b}_{\sigma(i)} - b_i\|_1 + \lambda_{\text{giou}} \mathcal{L}_{\text{giou}}(\hat{b}_{\sigma(i)}, b_i) \right), \tag{17}$$

where σ denotes the optimal bipartite matching. We keep the original language-box alignment losses in EDA, including the soft token loss and contrastive alignment loss, and associate each selected primitive box with the query text.

At inference time, EDA outputs a set of predicted AABBs rather than primitive labels directly. To convert its output into our binary primitive grounding result, we match each predicted AABB back to the closest B-Rep primitive AABB. Primitives matched by high-scoring predicted boxes are regarded as selected, and primitives are ranked according to the corresponding grounding scores. This allows EDA to be evaluated under the same primitive-level metrics as other baselines, while preserving its original box-grounding formulation.

C.2 BRepFormer

BRepFormer Dai et al. (2025) is originally designed for B-Rep geometric feature recognition. We adapt it to text-conditioned primitive grounding while keeping its original B-Rep backbone unchanged. Specifically, BRepFormer extracts geometric and topological features from the input B-Rep and produces face tokens together with a global B-Rep feature. Since the original model predicts face-level feature categories, we replace its recognition head with a primitive-level grounding head.

For edge grounding, we construct an edge token by combining the encoded edge feature with the representations of its incident faces. Let $\mathbf{h}_j^{\text{edge}}$ denote the encoded feature of edge e_j, and let $\mathbf{h}_i^f$ denote the face token of face f_i. We aggregate the incident face tokens with a permutation-invariant pooling operator and map the concatenated feature through an MLP ϕ_e:

$$\mathbf{h}_j^e = \phi_e \left([\mathbf{h}_j^{\text{edge}}; \mathrm{Pool}(\{\mathbf{h}_i^f \mid f_i \in \mathcal{N}(e_j)\})] \right), \tag{18}$$

where $\mathcal{N}(e_j)$ denotes the incident faces of edge e_j, $\mathrm{Pool}(\cdot)$ denotes mean pooling, and ϕ_e is a learnable projection MLP.

To introduce text conditioning, we encode the natural-language description with BERT and keep all text tokens. We then add cross-attention layers, where primitive tokens serve as queries and BERT text tokens serve as keys and values. The resulting text-conditioned primitive features are passed to a binary selection head:

$$\mathbf{s} = \mathrm{Head}_{\text{sel}} \left(\mathrm{CrossAttn} \left([\mathbf{H}^f; \mathbf{H}^e], \mathbf{T}^q \right) \right), \tag{19}$$

where $\mathbf{T}^q$ denotes the BERT text-token sequence. The model is trained with the same primitive-level labels as PGGM using focal loss over all face and edge primitives. This adaptation gives BRepFormer access to native B-Rep inputs and token-level text-primitive interaction, while leaving its original feature recognition backbone unchanged.

Table 6: Main model and optimization hyperparameters used in PGGM.

Category	Hyperparameter	Value
Model	Primitive embedding dimension	512
Model	Topology message-passing layers	6
Model	Prompt-conditioned fusion layers	8
Model	Selection decoder layers	6
Model	Attention heads	8
Model	Dropout	0.1
Model	GSP initial temperature τ	0.07
Model	BERT maximum length	256
Model	VAE LoRA rank	32
Model	VAE LoRA alpha	16
Model	VAE LoRA dropout	0.05
Model	BERT LoRA rank	32
Model	BERT LoRA alpha	16
Model	BERT LoRA dropout	0.05
Optimization	Optimizer	AdamW
Optimization	Initial learning rate	1×10^{-4}
Optimization	Weight decay	1×10^{-2}
Optimization	Warmup steps	100
Optimization	Gradient clipping	1.0
Optimization	Mixed precision	FP16 mixed precision
Loss	Focal loss α	0.75
Loss	Focal loss γ	1.0
Loss	Contrastive loss weight	0.2
Loss	Contrastive margin	0.3
Loss	Text-alignment loss weight	0.2
Loss	Text-alignment temperature	0.07
Loss	Consistency loss weight	0.1
Loss	Consistency activation threshold	$r \geq 0.80$

C.3 BRepMFR

BRepMFR Zhang et al. (2024) is adapted following the same protocol as BRepFormer. We retain its original B-Rep backbone and replace the machining feature recognition head with a primitive-level grounding head. Face and edge features are treated as primitive tokens and are further processed with the same text-conditioning and decoding pipeline, including cross-attention with BERT text tokens and a selection head for primitive-level prediction. The model is trained with the same primitive-level supervision and focal loss as in BRepFormer.

C.4 VLM-Verifier

The VLM-Verifier baseline is implemented as a rendering-based primitive verification pipeline using Gemini-2.5-Flash Google (2026). For each B-Rep model, all primitives are divided into groups of five. Each group is rendered from multiple views, with different primitives highlighted using distinct colors for identification. Given the rendered images and the natural-language description, the VLM is prompted to classify each highlighted primitive as either a positive or negative grounding target and to provide a confidence score for the decision. The binary predictions are used as primitive selection results, while the confidence scores are used to rank primitives for rank-based metrics such as R@K and mAP. All predictions are mapped back to the corresponding primitives and evaluated with the same primitive-level metrics as the other baselines.

D Hyperparameter Details

Table 6 summarizes the main model and optimization hyperparameters used in our experiments. Unless otherwise stated, all methods are trained under the same optimization setting for fair comparison.

For DB training, clean partial prompts are first used for warmup, after which training continues with a mixture of clean and noisy partial signals. For clean partial construction, we randomly sample

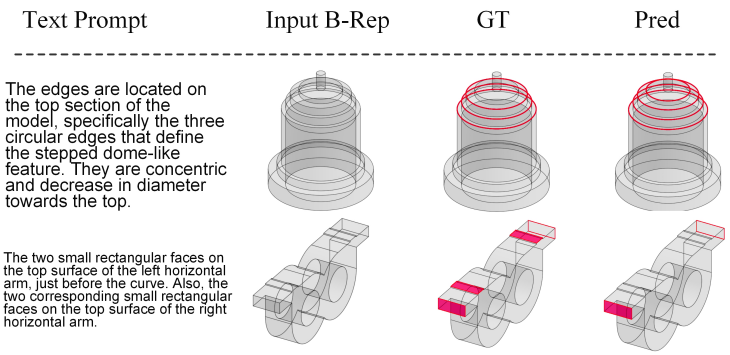


Figure 3: Representative failure cases of PGGM. The first case illustrates over-selection caused by an ambiguous text prompt, where nearby concentric edges are also plausible targets. The second case shows under-selection caused by incomplete semantic understanding of a multi-part spatial description.

10%–40% of positive primitives as partial hints. In SBD, the model selects the top-5 primitives from text-conditioned predictions, whereas in PCD, the hard-negative mining budget is set to 32. TID is constructed once after warmup and then kept fixed throughout training, while SBD is refreshed every epoch and PCD every two epochs.

E Resource Consumption

We report the resource consumption of both TID construction and model training for completeness. TID signals are constructed using Qwen3-VL-32B-Instruct Bai et al. (2025), with vLLM Kwon et al. (2023) for accelerated inference, requiring approximately 6 NVIDIA A100 GPU hours. Training a full PGGM model requires approximately 80 NVIDIA A100 GPU hours.

F Failure Cases

Figure 3 shows two representative failure cases. In the first case, the text prompt is semantically ambiguous: although it refers to "three circular edges," the surrounding description of a stepped dome-like feature is not sufficiently precise to uniquely separate the target rings from nearby concentric edges, causing the model to over-select plausible primitives. In the second case, the prompt specifies multiple small rectangular faces across two symmetric arms, but the model fails to fully capture the compositional spatial relation and selects only part of the intended targets. These cases suggest that future work should improve both ambiguity-aware grounding under underspecified language and fine-grained semantic reasoning over multiple related primitives.