CAD-RAG: A multi-modal retrieval augmented framework for user editable 3D CAD model generation

Ananthakrishnan A
Indian Institute of Technology, Madras
India, 600036, Chennai, Tamil Nadu
ananthu2014@gmail.com

Anush Bharathi
Madras Institute of Technology
India, 600036, Chennai, Tamil Nadu
anushbharathi2411@gmail.com

Dharanivendhan V
Indian Institute of Technology, Madras
India, 600036, Chennai, Tamil Nadu
dharanivendhanv01@gmail.com

Ramanathan Muthuganapathy
Indian Institute of Technology, Madras
India, 600036, Chennai, Tamil Nadu
emry01@gmail.com

ABSTRACT

Computer-Aided Design (CAD) has revolutionized design and manufacturing by enabling precise, complex models in collaborative environments. While similar CAD models with application-specific modifications are often required, designs are typically created from scratch due to challenges in retrieving existing models or generating editable ones. Although parametric CAD modeling has advanced through deep generative approaches treating CAD as a language task to generate user-editable designs, building truly scalable multi-modal datasets and networks tailored for 3D design tasks, particularly in engineering domains remains a significant challenge. Developing such datasets, especially those incorporating images, point clouds and user-like text and hand-drawn sketches is difficult as these modalities demand fine-grained geometric understanding and extensive human-in-the-loop evaluations. While large foundational models like CLIP have improved cross-modal retrieval, they are primarily trained on natural images and fail to capture the geometric and structural complexities inherent to CAD data.
In this paper, we propose a novel multi-modal pipeline for CAD command sequence generation using state-of-the-art Vision-Language Models (VLMs). We introduce a unique multimodal CAD dataset comprising hand-drawn sketches, CAD command sequences, images and basic text prompts. These modalities are integrated through a Multi-modal Retrieval-Augmented Generation (MM-RAG) framework to enable user-editable CAD model retrieval and generation. Our RAG-based pipeline streamlines the CAD design process by enabling iterative, user-guided model generation based on simple sketches or text queries. This approach aims to streamline CAD model design by creating an advanced, end-to-end pipeline that supports design workflows. The dataset and code will be made publicly available at: `https://github.com/ananthu2014/cadrag`.

Keywords
Computer Aided Design(CAD), 3D shape retrieval, Multi-modal dataset

1 INTRODUCTION

Computer-Aided Design (CAD) has been the torchbearer of modern design and manufacturing, transforming traditional workflows with greater precision, pace and efficiency. From small-scale 3D printed artifacts to large machinery and systems, the need for high-quality designs remains crucial. However, the development of skilled designers continues to be essential, necessitating investments in training to ensure that individuals are industry-ready to design using CAD software such as SolidWorks, Fusion 360 and others. With the advent of deep learning and advanced architectures, significant attention has been paid to data-driven approaches that help designers create better designs, with a primary focus on 3D representation learning for retrieval and generation of CAD models [1].

Retrieval systems focus primarily on searching and locating relevant 3D shapes in large databases through semantic/similarity matching. This includes point cloud-based and image-based [2] approaches among others. Given an input and a target modality, training aim to bring similar models closer in the embedding space and dissimilar ones farther apart. 3D model understanding lies at the crux of this problem, where models are trained to extract relevant features from the data and align them within a shared embedding space [3, 4, 5].

Earlier, content-based retrieval (CBR) systems (where relevant items are searched by analyzing their intrinsic features) employed rule-based techniques such as Poisson histograms [6] and Histograms of Orientation [7].

Model	Sketch	Text	Recall				MAP			
			k=1	k=2	k=5	k=10	k=1	k=2	k=5	k=10
Ours (SBIR)	×	✓	7.94	10.42	17.37	26.05	7.94	9.18	10.93	13.12
Ours (TBIR)	✓	×	14.14	22.08	34.74	47.39	14.04	18.11	21.68	23.37
Ours (STBIR)	✓	✓	**18.11**	**24.81**	**37.47**	**48.64**	**18.11**	**21.46**	**24.95**	**26.47**

Table 1: Zero-shot (Baseline) performance for Sketch-based (SBIR), Text-based (TBIR) and Sketch-Text based (STBIR) Image retrieval on our model measured by Recall and Mean Average Precision (MAP) at top-k retrieval for 403 test samples of CAD-RAG dataset.

With the development of learning-based approaches, useful features were extracted and learned from images, videos, sketches, text, point clouds and more, further advancing retrieval systems. Among these modalities, sketches and text are the most practical for user queries, particularly in the CAD domain, though point cloud could also be considered.

Significant developments have been made in sketch-based content retrieval, particularly in Sketch-Based Image Retrieval (SBIR) methods which mostly utilize dual-encoder Siamese networks to map a given sketch to its corresponding image(s) [8]. A major challenge in adopting this approach is the scarcity of large sketch-based CAD datasets corresponding to engineering shapes. Further, until the development of large pre-trained CLIP-like models [9], using text as a query was not feasible due to the unavailability of datasets that correlate text queries with other modalities to enable cross-modal retrieval.

Over the years, considerable attention has been directed toward geometric deep learning due to advancements in architectures capable of learning such complex representations. Many studies have focused on learning 3D representations from discrete forms. ComplexGen [10] reconstructs B-Rep models from point clouds, Sketch2Mesh [11] generates meshes from sketches while SDFusion [12] performs 3D reconstruction and completion in the form of Signed Distance Functions (SDFs) from multi-modal inputs such as images and text. Although these advancements enhance user control over the generation process, the resulting parametric representations remain non-editable, which is undesirable in a design workflow.

Further advances were introduced in DeepCAD [13], enabling the sequential generation of user-editable CAD models by treating modeling as a language-based task. Models such as Point2Cyl [14], Free2CAD [15], OpenECAD [16] and Text2CAD [17] support cross-modal generation, improving user control and bringing significant attention to parametric CAD generation.

With the advent of large models like CLIP [9], their application in self-supervised learning and adaptation to zero-shot downstream tasks has gained significant attention [18]. However, these models were trained on large-scale internet datasets, which differ significantly from the CAD domain leading to reduced zero-shot performance (see Table 1), particularly in text-to-image retrieval. This highlights the need for a comprehensive text dataset tailored to CAD.

Furthermore, the results reveal that the fine-grained nuances of engineering shapes make sketches a more expressive modality, achieving better performance compared to textual queries. This reinforces the need for a quality sketch dataset as well, one that mimics actual user queries. Additionally, large foundational models such as GPT [19] have demonstrated strong generalization capabilities in zero- and few-shot tasks, especially with retrieval-augmented generation (RAG) [20], suggesting promising avenues for CAD applications.

Hence, in this paper, a multi-modal dataset integrating text, point clouds, CAD command sequences, free-hand sketches and images is introduced, created through a combination of human-in-the-loop processes and deep learning methodologies leveraging SOTA foundational models for text and generative models for sketches. Furthermore, a novel RAG-based network is proposed, enabling sequential retrieval and refined generation of user-editable CAD models from simple user prompts. The entire pipeline is designed to be compute-efficient, with training conducted on low-end GPUs such as the NVIDIA RTX 3080 Ti and 4070 Ti.

The key contributions of this paper are:

- A one-of-its-kind multi-modal dataset, incorporating free hand-drawn sketches, command sequences, images, point clouds and 3-level text prompts based on DeepCAD[13].
- A novel multi-modal RAG pipeline, which is perhaps the first work in the field that performs sequential retrieval and generation of user-editable engineering/CAD shapes.

2 RELATED WORKS

2.1 CAD as a Language Task

Wu et al., in their work DeepCAD [13] proposed a Transformer-based autoencoder for the sequential generation of CAD models, treating CAD design similarly to a language task. To achieve this, a dataset was created from the Sketch-and-Extrude subset of the ABC Dataset [21] with a domain-specific language designed for sequential modeling. This was further improved by approaches such as Xiang Xu et al. [22, 23], which demonstrated high quality generation over the former.

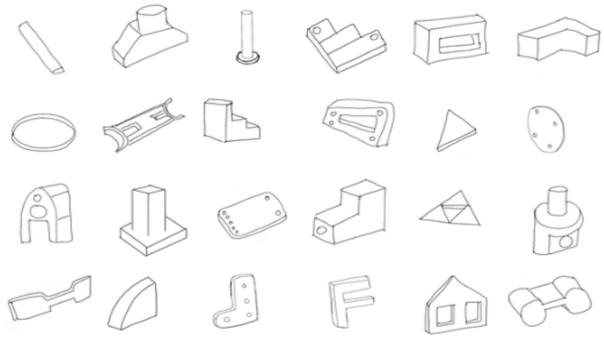


Figure 1: Samples from 403 hand-drawn sketches, drawn by 3 sketchers just by looking at the model

Furthermore, user-editable CAD models have been generated from point clouds [14], sequentially drawn vector sketches [15], images [16] and text prompts [17, 24], all utilizing the DeepCAD dataset. While image-based generation was showcased in OpenECAD [16], it relies on fine-tuned small Vision-Language Models (VLMs) with limited context windows and requires rendered images of CAD, making it a lesser feasible option for user queries.

Text2CAD [17] utilized Qwen2-VL-14B and Qwen2.5-72B-Instruct [25] to generate textual descriptions for the models of DeepCAD dataset. Twenty CAD image views per model, along with CAD operation sequences extracted from metadata, were used as input. The generation process was performed on multiple A100 GPUs over several days, requiring significant computational resources and resulting in a dataset that is inferior to ours which was created using SOTA VLM Gemini [26], as discussed in a later section. Additionally, the dataset was not publicly available at the time of our research.

2.2 Sketch and Text-based 3D Retrieval

Most research on sketch-based representation learning has focused on the vector level [27, 28] due to the sparse nature of pixel-level sketches and the unavailability of large datasets, particularly in the engineering domain. CADSketchNet [29] introduced a dataset having 58,000 sketches based on the MCB [30] dataset using Canny edge detection. However, quality issues such as mimicking exact edges and including unwanted mesh lines reduced its realism compared to user-drawn sketches, even though attempts have been made to improve the quality [31, 32]. Additionally, most existing works have focused on supervised datasets, where the performance is evaluated based on retrieval from the same class.

However, no prior work has explored text-based CAD model retrieval due to the lack of large, high-quality datasets. Most existing work primarily focuses on CAD generation using deep generative models, as discussed earlier. Multi-modal retrieval based on freehand sketches and text prompts requires significant attention, given its practical advantages for real-world designers to search for relevant 3D models. Creating such datasets requires substantial time and human involvement, especially to build large-scale, human-like collections suitable for unsupervised learning.

2.3 Point-Cloud representation learning

3D point clouds effectively represent CAD models by capturing their full 3D structure in $\mathbb{R}^3$. Additionally, real-world user queries can be performed using scanned point clouds obtained from devices like LiDAR and Microsoft Kinect, making it a viable domain for users. Models such as PointNet and DGCNN [4, 5] effectively extract global and local features from point clouds and can be adapted for downstream tasks.

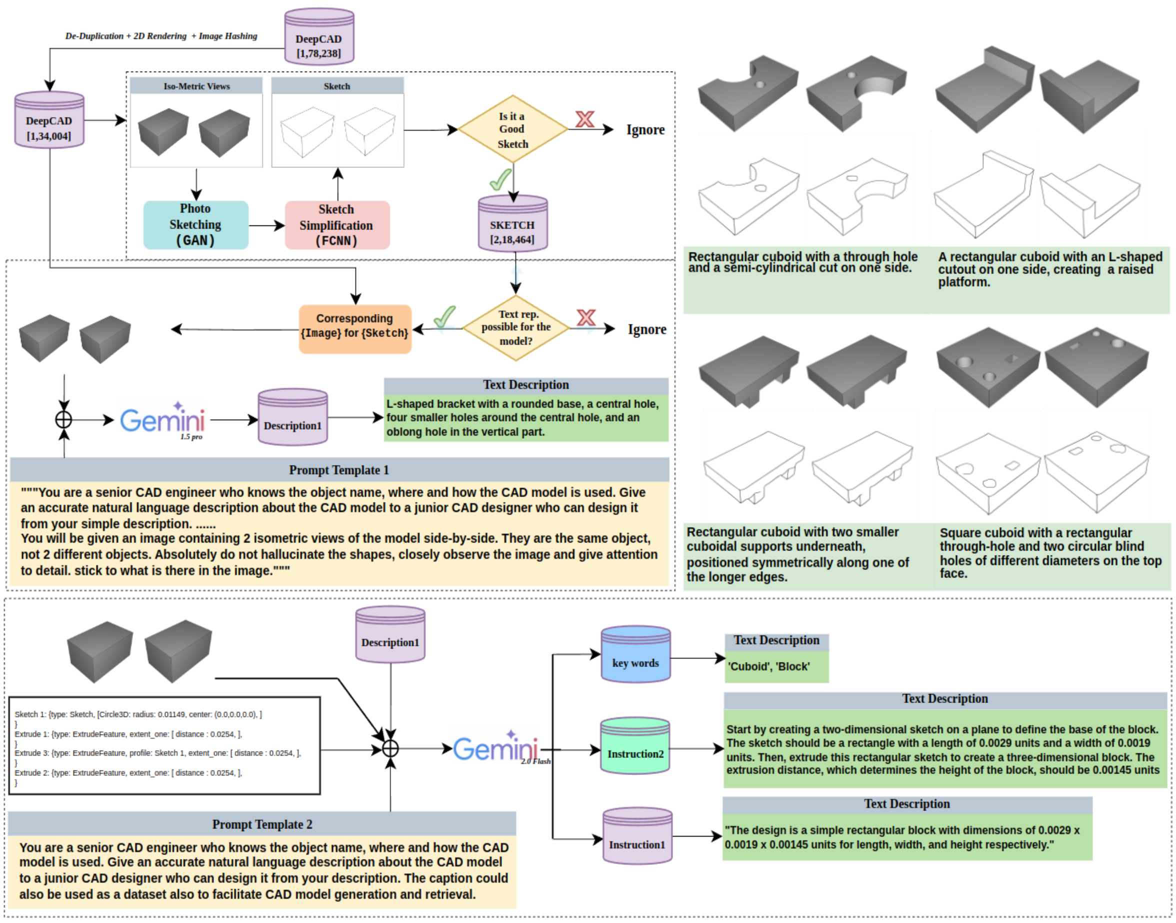
Recently, multi-modal learning has gained traction, utilizing large models to align different modalities such as point clouds, text and images thereby facilitating downstream tasks such as classification, retrieval and segmentation [33, 34]. However, these models are typically trained on generic supervised datasets like ShapeNet [35] and ModelNet [36].

2.4 Retrieval-Augmented Generation

Retrieval Augmented Generation (RAG) enhances the performance of LLMs by integrating relevant knowledge during generation, improving factual accuracy and access to necessary information [20]. Multi-Modal RAG extends this framework by leveraging both textual and visual inputs to produce richer, context-aware outputs with reduced hallucinations. Existing CAD sequence generation methods rely on large GPU training and dataset-specific representations, limiting scalability. While recent approaches like OpenECAD [16] demonstrate the benefits of foundational models such as GPT [19] for few-shot generation, to the best of our knowledge there are no existing works that utilize Multi-Modal RAG for improving CAD generation.

3 DATASET CREATION

We used CAD models from the DeepCAD [13] dataset, originally consisting of 178,238 models. After deduplication following the approach of Willis et al. [13], 137,004 unique models were obtained. These were used to create a comprehensive dataset encompassing freehand-drawn sketches, text captions, point clouds and isometric images, integrated with CAD construction sequences from OpenECAD dataset [16].



Rectangular cuboid with a through hole and a semi-cylindrical cut on one side.

A rectangular cuboid with an L-shaped cutout on one side, creating a raised platform.

Rectangular cuboid with two smaller cuboidal supports underneath, positioned symmetrically along one of the longer edges.

Square cuboid with a rectangular through-hole and two circular blind holes of different diameters on the top face.

L-shaped bracket with a rounded base, a central hole, four smaller holes around the central hole, and an oblong hole in the vertical part.

"""You are a senior CAD engineer who knows the object name, where and how the CAD model is used. Give an accurate natural language description about the CAD model to a junior CAD designer who can design it from your simple description. ......
You will be given an image containing 2 isometric views of the model side-by-side. They are the same object, not 2 different objects. Absolutely do not hallucinate the shapes, closely observe the image and give attention to detail. stick to what is there in the image."""

```
Sketch 1: {type: Sketch, [Circle3D: radius: 0.01149, center: (0.0,0.0,0.0), ]
}
Extrude 1: {type: ExtrudeFeature, extent_one: [ distance : 0.0254, ],
}
Extrude 3: {type: ExtrudeFeature, profile: Sketch 1, extent_one: [ distance : 0.0254, ],
}
Extrude 2: {type: ExtrudeFeature, extent_one: [ distance : 0.0254, ],
}
```

You are a senior CAD engineer who knows the object name, where and how the CAD model is used. Give an accurate natural language description about the CAD model to a junior CAD designer who can design it from your description. The caption could also be used as a dataset also to facilitate CAD model generation and retrieval.

'Cuboid', 'Block'

Start by creating a two-dimensional sketch on a plane to define the base of the block. The sketch should be a rectangle with a length of 0.0029 units and a width of 0.0019 units. Then, extrude this rectangular sketch to create a three-dimensional block. The extrusion distance, which determines the height of the block, should be 0.00145 units

"The design is a simple rectangular block with dimensions of 0.0029 x 0.0019 x 0.00145 units for length, width, and height respectively."

Figure 2: The complete pipeline for data set creation: sketch (top) and text (bottom) with data samples showcasing multi-modal prompts (right).

3.1 Image and Point cloud Data

Using the Python-based rendering tool VTK, different views of the CAD models were rendered under ambient conditions from various isometric viewpoints. After human inspection of random samples, two viewpoints with directional cosines $\left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)$ and $\left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)$ were finalized to maximize model coverage. Image hashing was further applied to remove strict duplicate models from the dataset to some extent. Further, the point cloud dataset was generated using `Trimesh`, consisting of samples with 8,192 and 4,096 points obtained via Farthest Point Sampling (FPS), with added noise to facilitate improved generalization.

3.2 Sketch Data

Creating a large-scale dataset of hand-drawn sketches at scale of hundreds of thousands is extremely labor-intensive and time-consuming. Given the absence of existing large-scale sketch datasets, we utilized a deep learning-based pipeline inspired by M. Li et al. [37] and Simo-Serra et al. [38] for creating a larger dataset. Specifically, the Photo-Sketching method [37] based on Conditional GAN architecture was employed to generate pixel-level, human-like contour sketches from images. A fully convolutional neural network [38] was then sequentially integrated into this pipeline to enable rough cleanup and refinement of the sketches produced by the initial GAN-based model.

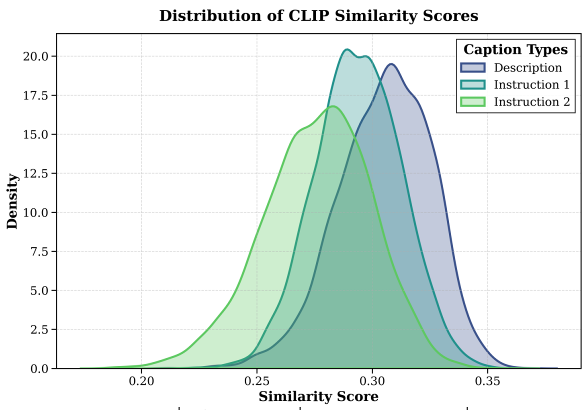
For fine-tuning these models, 750 hand-drawn sketches were manually created by roughly tracing over images of selected CAD models from the DeepCAD [13] dataset. Three sketchers were employed for this task, with each sketcher given a maximum of 25 seconds per image. Diverse images across different views were manually selected to capture the full data distribution.

After training and then generating sketches for the entire image dataset, a human-in-the-loop review was conducted, where each reviewer rated the sketches as either ’Good’ or ’Bad’ based on reference images. Models with incorrect view angles were removed. The final dataset includes two sketches per model for 109,232 models (refer to Figure 2 (top)), with 27,772 sketches labeled as ’Bad Sketches’ and removed.

Additionally, 403 sketches were drawn solely by looking at the models, creating a real-case test dataset that avoids pixel-to-pixel correspondence with the original rendered images and better mimics actual user input (refer to Figure 1). Three sketchers were given a maximum of one minute per sketch.

3.3 Text Data

Text2CAD [17] and CADTranslator [39] employed large VLMs and LLMs such as Qwen [25], Mistral [40] and CoCa [41] to generate text prompts via image captioning. In Text2CAD, four levels of captions, from beginner to expert, were created with the process demanding substantial computational resources.



Caption Type	Average Tokens	Mean Image Similarity	Std dev.
Description	14.09	0.3052	0.0202
Instruction-1	27.26	0.2936	0.0187
Instruction-2	74.08	0.2762	0.0234

Figure 3: Text-image similarity analysis using a pre-trained CLIP model, comparing different types of captions to evaluate alignment with image representations.

To address these challenges, a cost-effective pipeline is proposed, utilizing Gemini 1.5 Pro and Gemini 2.0 Flash [26], which are among the state-of-the-art VLMs available. Unlike in Text2CAD, where a VLM is employed in the first stage and an LLM in the second, a two-stage generation process using VLMs at both stages is adopted in our approach.

Additionally, Gemini 1.5 Pro costs **\$0.10 per 1M** input tokens and **\$0.40 per 1M** output tokens, making API-based caption generation both computationally and economically efficient thereby eliminating the need for heavy compute resources. Instead of the 20 viewpoints used by Text2CAD, only two isometric CAD images were utilized, as the DeepCAD [13] models are predominantly symmetrical. Through careful prompt engineering, captions were refined to be more human-like, with special attention given to avoid unnecessary explanations, ensuring that the prompts remained concise yet detailed and avoiding references to terms such as "CAD," "images," or color descriptions.

Since CLIP [9]-like models has a maximum token limit of 77 for text input, care was taken to maintain a lower average token count while preserving the geometric details. The overall procedure for text-data generation is outlined as follows:

Level 1 (CAD-RAG Description): Preliminary descriptions for the models were generated using Gemini-1.5 Pro, where two isometric views per CAD model were provided to the VLM along with a structured prompt to generate the captions.

Level 2 (Instruction 1 and 2): To enhance complex reasoning, the Gemini-2.0 Flash model was used. The images, preliminary description and a revised prompt were provided to generate three levels of captions: **Instruction 1 (Crisp)**, **Instruction 2 (Elaborate)** and a set of keywords. To facilitate the accurate generation of design instructions, CAD construction sequences, along with dimensions were extracted from the Metadata/FeatureScript format of DeepCAD dataset models obtained from Onshape and incorporated with the images and descriptions. Zero-shot image similarity analysis using CLIP [9] demonstrated that the Level-1 description exhibited higher text-image similarity compared to the other two caption levels, which will further aid retrieval performance and was therefore utilized for the training of our model. Figure 3 shows the comparison between different caption levels based on average token count and average zero-shot image-text similarity evaluated on a pre-trained CLIP model.

Subsequently, the above dataset was integrated with the CAD construction sequences proposed by OpenECAD[16] to form the final dataset. Figure 2 depicts the entire pipeline of dataset creation and showcases some excerpts from the dataset.

4 METHODOLOGY

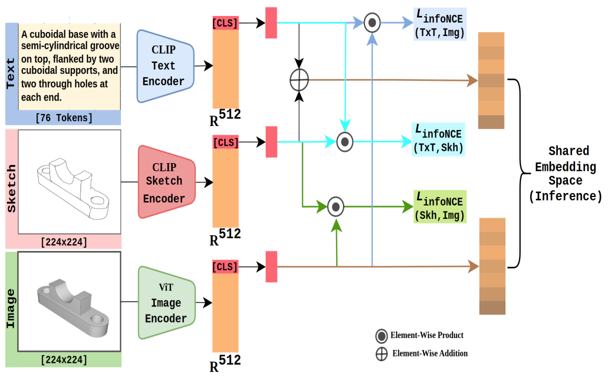


Figure 4: Final Architecture of Sketch and Text-based Image-Retrieval(STBIR)

The proposed framework, CAD-RAG, comprises two parts: **Retrieval** and **Generation**. Based on user prompts consisting of sketches (S_i) and/or text (T_i), the network retrieves the top-k ranked images (I_i) or point

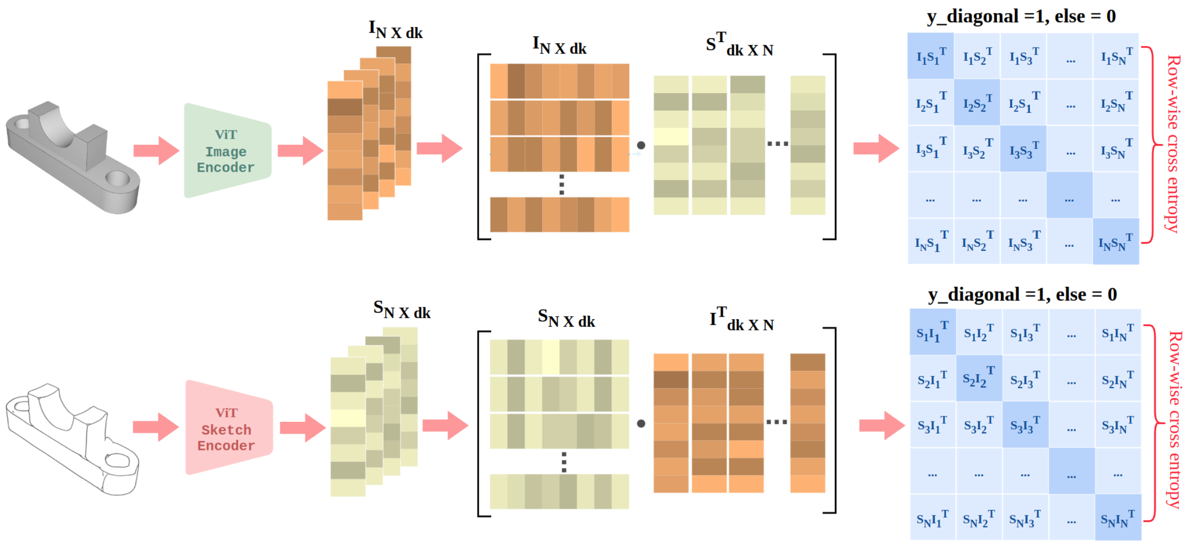


Figure 5: Depiction of Symmetric Contrastive learning (InfoNCE) loss

clouds (P_i) of CAD models from the saved embeddings. The corresponding CAD command sequences (CS_i) are stored as hash maps $\{\mathrm{I/P}_i : \mathrm{CS}_i\}$, enabling retrieval based on text or sketch queries. The best retrieval method is selected based on experimental results and is subsequently used to assist the generation process.

In the generation phase, the retrieved construction sequences serve as few-shot samples, which then combined with user prompts to enable the VLM to generate the final refined model. Each training pair is represented as $(S_i \cup T_i, I_i/P_i : CS_i)$.

4.1 Retrieval

Image-Based Retrieval:

The network architecture is shown in Figure 4. Given a user query (sketch and/or text), the network uses vision and/or text encoders to generate embeddings, which are compared with those from a database of image embeddings that shares the same latent space. The most similar results are then retrieved, similar to a search engine.

Contrastive Learning with InfoNCE Loss:

Since the dataset is unlabeled, traditional triplet loss requiring explicit positive and negative samples based on class labels is not applicable. Instead, a self-supervised learning approach was adopted using an in-batch symmetric contrastive learning strategy, following the Noise-Contrastive Estimation (InfoNCE) loss formulation to align multimodal embeddings. This loss encourages semantically similar samples from different modalities to be mapped close to each other in the embedding space, while keeping away the dissimilar ones in the embedding space.

The InfoNCE loss is defined as:

$$\mathscr{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\mathrm{sim}(z_i, z_i^+)/\tau)}{\sum_{j=1}^{N}\exp(\mathrm{sim}(z_i, z_j)/\tau)}$$

where z_i and z_i^+ denote the embeddings of a positive pair, $\mathrm{sim}(\cdot,\cdot)$ is the similarity function (e.g., cosine similarity), and τ is a temperature hyperparameter controlling the distribution sharpness.

Each sample consists of a pair from two modalities, such as (image$_i$, sketch$_i$) or (image$_i$, text$_i$), with $i = 1, 2, \ldots, N$ and batch size N. A batch thus includes positive pairs $\{(i_1, s_1), (i_2, s_2), \ldots, (i_N, s_N)\}$, where i_k and s_k are matched image and sketch embeddings.

During training, for a given anchor z_i (e.g., embedding of image$_i$), the corresponding positive is z_i^+ (e.g., sketch$_i$), and the remaining $N-1$ sketch embeddings $\{z_j\}_{j\neq i}$ serve as negatives. Hence, a batch yields N positive and $N(N-1)$ negative pairs. Self-pairing (e.g., image$_i$ with image$_i$) is excluded from the loss. Figure 5 illustrates the loss computation.

The overall methodology is summarized as:

- **Image Encoder:** A Vision Transformer (ViT) [42] extracts embeddings $I_f \in \mathbb{R}^{N\times d_i}$ from images.

- **Text/Sketch Encoder:** A Transformer model (BERT [43] or ViT [42]) extracts embeddings $T_f \in \mathbb{R}^{N\times d_t}$, with $d_i = d_t$.

- **Similarity Computation:** Pairwise cosine similarities are computed and scaled by a learnable temperature t as logits $= I_e T_e^\top \times \exp(t)$.

- **Loss formulation:** A symmetric cross-entropy loss is applied over similarity logits, averaging losses from both Image $\rightarrow$ Text/Sketch and Text/Sketch $\rightarrow$ Image directions. Since cross-entropy operates row-wise, symmetric training ensures balanced alignment across modalities and improves retrieval.

Using this, we propose a tri-modal loss function:

$$\mathscr{L}_{\text{total}} = \mathscr{L}(I,S) + \mathscr{L}(I,T) + \mathscr{L}(T,S)$$

where I, S, and T correspond to images, sketches and text respectively. Each $\mathscr{L}$ term denotes the symmetric InfoNCE loss computed between the respective pairs.

During inference, if the user query includes both modalities, the features extracted from the text and sketch encoders are averaged to obtain a unified embedding in the common representation space.

Point Cloud-Based Retrieval:

Given a user query (sketch or text), the network encodes it using fine-tuned vision and text encoders and retrieves the most similar embedded point cloud vector.

Sketch-based:

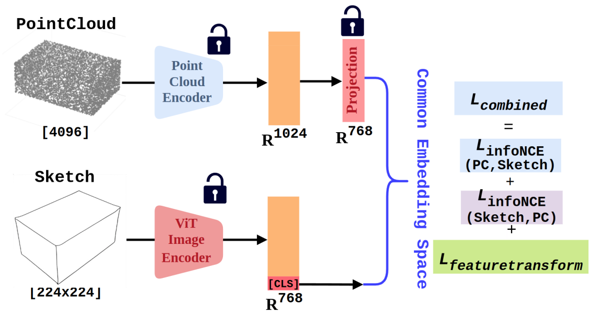


Figure 6: Final architecture of sketch-based point cloud retrieval, which achieved the best performance among all configurations.

The point cloud and sketch encoders are jointly fine-tuned to align their embeddings in a shared latent space using the symmetric InfoNCE loss. A projection layer is added after the point cloud encoder to map features to a common dimension. When using PointNet [4], a feature transform loss is also applied to ensure affine invariance and improve stability. Refer to Figure 6 for the final architecture.

The total loss used during this stage is defined as:

$$\mathscr{L} = \mathscr{L}^{s-p}_{\text{InfoNCE}} + \mathscr{L}^{p-s}_{\text{InfoNCE}} + \mathscr{L}_{\text{FT}} \qquad (1)$$

where $\mathscr{L}^{s-p}_{\text{InfoNCE}}$ aligns the sketch embeddings with point cloud embeddings, $\mathscr{L}^{p-s}_{\text{InfoNCE}}$ enforces reverse alignment from point clouds to sketches and $\mathscr{L}_{\text{FT}}$ denotes the feature transform regularization loss.

Text-based: Unlike the sketch modality, learning an effective text-based retrieval system is more challenging due to the relatively limited information content in text prompts, especially given the nuanced geometric nature of engineering shapes. Moreover, the contour-based sketch data exhibits higher visual quality and geometric fidelity, whereas the text descriptions tend to be less descriptive and less structurally informative.

The best performing model was trained on three stages:

I: Training Point Cloud Auto-Encoder: A PointNet-Mini [4] based auto-encoder is trained to reconstruct input point clouds using Chamfer Distance as the loss:

$$\mathscr{L}^{p-p}_{\text{Chamfer}} = \sum_{x\in P} \min_{y\in Q} \|x-y\|^2 + \sum_{y\in Q} \min_{x\in P} \|x-y\|^2$$

where P and Q are the input and generated point clouds.

II: Training Text Encoder with frozen Point Cloud module: The text encoder is trained while freezing the point cloud encoder and decoder. The loss combines Chamfer loss and symmetric InfoNCE alignment:

$$\mathscr{L}_{\text{stage2}} = \mathscr{L}^{t-p}_{\text{Chamfer}} + \mathscr{L}^{t-p}_{\text{InfoNCE}} + \mathscr{L}^{p-t}_{\text{InfoNCE}}$$

The Chamfer loss ensures accurate reconstruction of the point cloud from the text embedding. Given the text encoding, the decoder reconstructs the point cloud, and the Chamfer loss is computed between the reconstructed and original point clouds. Refer to Figure 7 for the architecture of Stage I and II training.

III: Joint Fine-Tuning: Both text and point cloud encoders are fine-tuned jointly using symmetric InfoNCE loss. This stage mirrors the architecture and training setup used in sketch-based point cloud retrieval.

$$\mathscr{L}_{\text{stage3}} = \mathscr{L}^{t-p}_{\text{InfoNCE}} + \mathscr{L}^{p-t}_{\text{InfoNCE}}$$

During inference, the sketch/text query is encoded and matched against the database of encoded point-cloud embeddings and the most similar model is retrieved.

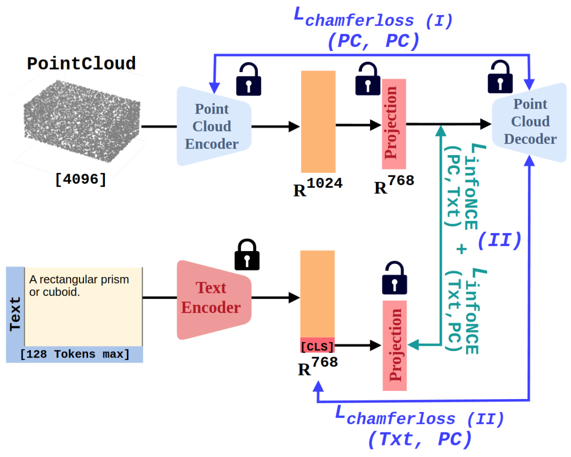


Figure 7: Overview of Stage I and II in text-based point cloud retrieval. Different loss functions are applied at each stage and denoted using I and/or II to indicate their respective stages.

4.2 Generation of editable CAD models

The performance of models proposed in OpenECAD [16] are used as the baseline for comparison, where VLMs are trained using the methodology

Model	Sketch	Text	Projection	Frozen I/P Enc.	Epochs	Recall			MAP		
						$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$
ViT B/16 - PointNet-Mini	✓	×	✓	✓	50	34.65	61.7	71.95	34.65	43.84	46.33
ViT B/16 - PointNet-Mini	✓	×	✓	×	50	71.55	90.5	94.3	71.55	78.1	79.19
ViT B/16 - PointNet-Mini	✓	×	×	×	50	70.35	90.5	95.25	70.35	77.58	78.78
ViT B/16 - PointNet	✓	×	✓	✓	50	46.55	73.15	82.5	46.55	55.73	57.99
ViT B/16 - PointNet	✓	×	✓	×	50	72.15	93.25	96.55	72.15	79.72	80.7
ViT B/16 - PointNet	✓	×	×	×	50	74	93.7	93.15	74	81.11	82.06
ViT B/16 - DGCNN	✓	×	✓	✓	50	57.5	84	90.55	57.5	66.42	68.27
ViT B/16 - DGCNN	✓	×	✓	×	50	77.45	94.05	97.4	77.45	83.36	84.28
ViT B/16 - DGCNN	✓	×	×	×	50	**78.95**	**94.75**	**97.55**	**78.95**	**84.54**	**85.32**
BERT - PointNet	×	✓	✓	✓	30	4.3	16.1	26.9	4.3	6.27	9.73
BERT - DGCNN	×	✓	✓	✓	30	5.2	16.8	27.6	5.2	9.06	10.47
BERT - PointNet-Mini	×	✓	✓	✓	50 + 30 + 30	**6.5**	**18.2**	**28.4**	**6.5**	**10.49**	**11.42**

Table 2: Performance comparison of various models on sketch/text-based point cloud retrieval on Validation data. The table shows different configurations of input encoders and point cloud encoders, with performance evaluated using MAP and Recall metrics at different k values. Best-performing models for each category are highlighted.

proposed in TinyLLaVA [44] to generate Python-level CAD operation sequences from input CAD images. However, their generalization to modalities such as hand drawn sketches and text prompts is found to be limited due to pretraining on different representation and the absence of multi-modal context. Furthermore, the Vision-Language Models VLMs) employed in these models are constrained by relatively small context windows (maximum 3072 tokens), often leading to incomplete generations.

To address these limitations, the feasibility of a Retrieval-Augmented Generation (RAG) framework is explored. In this approach, the most similar CAD construction sequence is retrieved based on user prompts (sketch and/or text) and used to guide the generation of an accurate CAD command sequence using large foundational models without fine-tuning, as illustrated in Figure 9. The CAD construction sequence format follows that of OpenECAD [16]. Foundational models such as GPT-4o Mini, Gemini-2.0 Flash, and Gemini-1.5 Flash [26, 19] were employed, leveraging their few-shot learning capabilities and large context windows. The performance of the proposed approach is compared against OpenECAD [16] models.

A structured template containing instructions on the command sequence format and operations, along with user prompts and the top-1 retrieved result (command sequence) was provided as input to iteratively refine the retrieved sequence and generate the desired one. In the case of RAG, the top-1 result was used, whereas a random model or no model was provided in the other case.

5 ABLATION STUDIES & RESULTS

5.1 Retrieval Performance:

Retrieval performance was studied for both image and point cloud as targets, where input modalities included freehand sketches and text prompts. A total of 10,000 models were randomly sampled from the dataset for training and comparison studies. The train-validation split was 9000-1000 and the 403 hand-drawn samples and its corresponding text captions served as the test set, which was not a part of the train-val data. Among the three-level captions, the Level 1 description exhibited better text-image similarity compared to the others while also having a lower token count. Therefore, it was used as the text data along with two sketches and images per model for analysis (Refer to Figure 3).

For point cloud retrieval based on sketch or text, BERT [43] was used as the text encoder in all experiments, while ViT-B/16 [42] served as the sketch encoder. The models were trained on two NVIDIA RTX 3080 Ti GPUs for varying numbers of epochs. DGCNN [5] models were trained with a batch size of 8, PointNet [4] with a batch size of 16 and PointNet-Mini [4] with a batch size of 32, using the Adam optimizer with a learning rate of 1×10^{-4} and 4096 points. Refer to Table 2 for the training configurations.

For image-based retrieval, a tri-modal InfoNCE loss was used along with the AdamW optimizer and a cosine annealing scheduler, with a learning rate of 2×10^{-5} and a weight decay of 1×10^{-5}. The model was trained on an NVIDIA RTX 4070 Ti for 15 epochs. Some sketch views were randomly dropped out during training to prevent over-fitting. These hyperparameters were selected based on best practices established in CLIP [9].

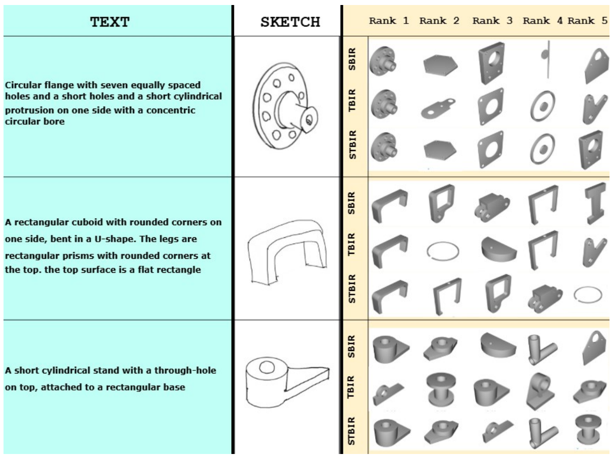

TEXT
SKETCH
Rank 1
Rank 2
Rank 3
Rank 4
Rank 5
Circular flange with seven equally spaced holes and a short holes and a short cylindrical protrusion on one side with a concentric circular bore
A rectangular cuboid with rounded corners on one side, bent in a U-shape. The legs are rectangular prisms with rounded corners at the top. the top surface is a flat rectangle
A short cylindrical stand with a through-hole on top, attached to a rectangular base
SBIR
TBIR
STBIR

Figure 8: Top-5 rank retrieval results for Sketch-based (SBIR), Text-based (TBIR) and Sketch+Text-Based Image Retrieval (STBIR) models.

Model	Sketch	Text	Recall			MAP		
			$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$
ViT-Siamese (ViT-B/16)	✓	×	92.25	98.05	99.70	92.25	95.15	95.66
DenseNet - Siamese	✓	×	84.80	93.20	98.65	84.80	89.00	90.54
ResNet - Siamese	✓	×	83.05	91.65	97.50	83.05	87.35	89.05
CLIP- (ViT-B/32)	×	✓	43.95	56.15	71.35	43.95	50.05	54.28
Ours (STBIR)	✓	✓	**92.70**	97.00	99.40	**92.70**	94.25	94.95
ViT-Siamese (ViT-B/16)	✓	×	52.61	67.00	77.67	52.61	59.80	62.68
Ours (SBIR)	✓	×	62.78	85.61	93.05	62.78	71.91	72.93
Ours (TBIR)	×	✓	56.82	85.86	92.56	56.82	68.21	69.15
Ours (STBIR) + L_{avg}	✓	✓	75.93	86.85	95.29	75.93	81.39	83.05
Ours (STBIR) + $L_{tri\text{-}modal}$	✓	✓	**77.92**	**96.28**	**98.26**	**77.92**	**85.45**	**85.71**

Table 3: Performance comparison of various models on validation and test data. The first section represents results on the validation set (contour sketches), while the last four rows indicate performance on the test set.

Performance was evaluated using Recall and Mean Average Precision (MAP) at top-k ranks ($k = 1, 5, 10$) on validation data for point cloud retrieval and on both validation and test sets for image retrieval. Recall@k measures the fraction of relevant CAD models retrieved in the top k and MAP evaluates the quality of ranking by averaging precision at each relevant retrieval position across queries, providing a comprehensive assessment of retrieval effectiveness. For multi-view sketches, metrics are counted only when the exact view is retrieved.

The best-performing models for text and sketch-based point cloud retrieval were trained using a three-stage and a single-stage pipeline respectively as described earlier (see Figure 6 and Figure 7). For text-based retrieval, PointNet-Mini [4] outperformed PointNet [4] and DGCNN [5], despite having fewer parameters, due to the multi-stage training. For sketch-based retrieval, the ViT-DGCNN model achieved the best performance.

Ablation studies revealed that projection layers improved text-based retrieval but slightly degraded sketch-based performance. Similarly, freezing the input encoder benefited text-based, while fine-tuning was more effective for sketch-based retrieval. Although sketch-based results were good, the performance of text-based retrieval remained notably poor.

The results suggest that the image-based retrieval model performs significantly better than the point cloud-based model, demonstrating stronger generalization to real user queries and achieving higher performance on the validation data in both MAP and Recall. Experiments with sketch-only (SBIR), text-only (TBIR) and combined sketch-text (STBIR) inputs showed the best performance in the combined case. An average loss $\mathscr{L}_{avg}$, computed using symmetric InfoNCE on averaged sketch and text embeddings was also tested, but the tri-modal loss outperformed it. Refer to Table 3. Figure 8 shows SBIR, TBIR, and STBIR retrieval examples on the test set.

Furthermore, on the validation set, the ViT-Siamese model was observed to perform slightly better than the STBIR model at $k = 5$ and $k = 10$. This can be attributed to the well-articulated contour sketch data in the validation set, whereas the inclusion of relatively weaker text descriptions slightly degraded retrieval performance in the STBIR setup.

5.2 Generation:

Experiments were conducted using the Gemini-1.5-Flash, Gemini-2.0-Flash [45], GPT-4o and GPT-4o Mini [19] models with and without RAG. Prompts were designed to focus the model's attention on the geometry itself, as user inputs such as sketches and text in this approach do not convey dimensional information. The results were compared with the pre-trained version of OpenECAD's 3.1B model [16].

The test dataset prompts (sketch and text) were provided to the models along with the top-1 retrieved Python-level API CAD construction sequence. The retrieval results were obtained using the best-performing STBIR model. To prevent exact model matching and better simulate real-world scenarios, retrieval was performed from the 10,000-model dataset, which does not include the exact models used for queries. Refer to Figure 9 (left) for the complete RAG pipeline.

Accuracy calculations are performed based on OpenECAD's evaluation metrics [16], using curve accuracy, loop accuracy and a weighted average score. In addition, we used the median CLIP score to compare the similarity between the generated models and the ground-truth models [16].

Median CLIP Score: Measures the similarity between the generated and the ground-truth CAD models. The isometric views of the generated and ground truth models are rendered and the similarity score is computed using a pre-trained CLIP [9] image encoder.

Curve Accuracy: (acc$_c$) Evaluates how well individual curves in the generated sketch match the reference, based on type, order and connectivity. Computed as the fraction of correctly generated curves.

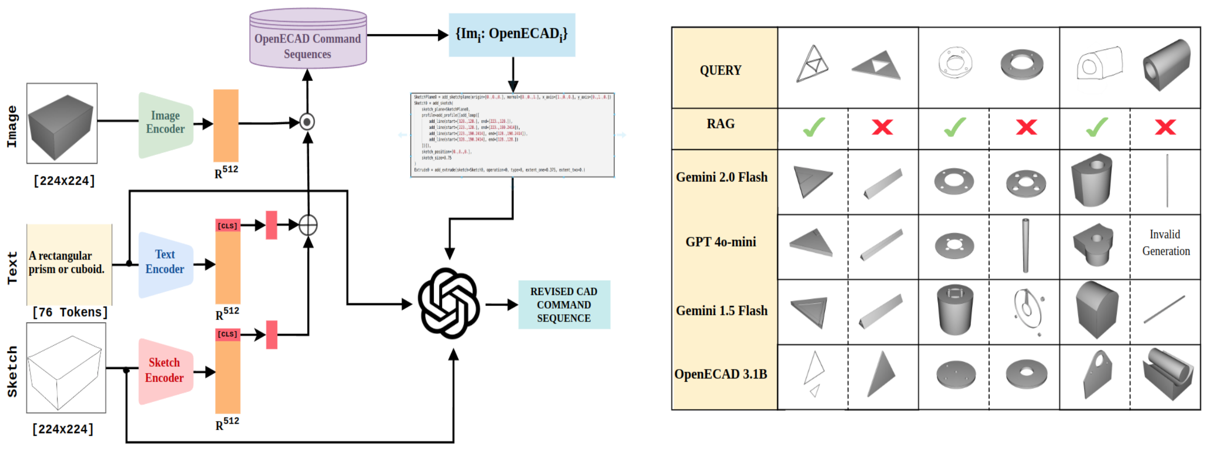


Figure 9: Complete pipeline for CAD command sequence generation using RAG (left) and example results of CAD generation using different models with and without RAG (right).

Loop Accuracy: (acc$_l$) Measures how accurately the generated loops match the ground truth in count and structure. Fully correct loops receive a score of 100; otherwise, it is weighted based on curve accuracy.

Weighted Average Score (WAS): Combines executability, curve accuracy, and loop accuracy to provide an overall evaluation score, computed as:

$$\text{WAS} = 10e + 45\text{acc}_c + 5\text{acc}_l + \frac{40}{L}\sum_{i=1}^{L}(10s_i + 90\text{acc}_i)$$

where e is executability, acc$_c$ is curve accuracy, acc$_l$ is loop accuracy, L is the number of loops and s_i, acc$_i$ represent correctness and accuracy of the i-th loop.

The results shown in Table 4 indicate performance improvements when the RAG approach was employed on Large Foundational Models compared to OpenECAD [16] models, with particularly better outcomes observed for Gemini-Flash-2.0. The median CLIP score was not calculated for the OpenECAD models because many of them were non-executable.

The models were tested under various scenarios to determine the optimal setup. When RAG was not employed, performance was notably worse, particularly when only the OpenECAD Python-API code operations were provided as context. In such cases, the generated outputs often failed to satisfy dimensional constraints, as the sketch and text inputs lacked dimensional informations. Results improved when a random model was supplied and further improved with the use of RAG.

The lower accuracy observed for OpenECAD 3.1B compared to other VLMs can be attributed to its relatively smaller size and pre-training on CAD images, leading to a lack of generalization to the sketch domain. Additionally, the context window length of the best-performing OpenECAD [16] model is limited to 3072 tokens, resulting in token exhaustion during RAG. This leads to incomplete and non-executable CAD construction sequences and a lower weighted average score. On the other hand, models such as GPT and Gemini, which have larger context windows, effectively attended to all input prompts and generated results more efficiently without requiring fine-tuning. Refer to Figure 9 (right) for the qualitative results with and without RAG for a given model.

6 CONCLUSION & FUTURE WORK

We introduced a novel RAG pipe-line for generating user-editable CAD models, leveraging an in-house developed multimodal dataset. We evaluated the proposed approach using actual user prompts, confirming its capability to support accurate CAD generation through effective retrieval. The hardware-friendly nature makes it particularly suitable for widespread deployment on consumer-grade PCs, enabling broader accessibility.

The model was developed and trained solely on the DeepCAD [13] dataset, which comprises relatively simpler models. Since training used contour sketches, generalization to real hand-drawn sketches is limited. While the text prompts were generated using state-of-the-art VLMs, additional human refinement is needed for full accuracy. Moreover, despite its advantages, CAD command sequence generation is not a scalable solution unlike retrieval, which can be extended to various datasets and modalities. Therefore, greater emphasis was placed on the latter.

Future work could explore fine-tuning smaller models (SLMs) to enhance domain-specific performance. Few-shot learning may offer improvements over the current one-shot setup, and further ablation studies could refine parameter choices, particularly in the generation stage. Iterative generation with a feedback loop also holds potential for improving accuracy. Additionally, expanding the dataset by incorporating scanned point clouds and more hand-drawn sketches would enhance its realism and better reflect practical use cases.

Model	RAG	Sketch	Text	Curve Acc (%)	Loop Acc (%)	Weighted Avg Score (%)	Median Clip Score
OpenECAD 3.1B	×	✓	×	32.07	35.73	57.84	-
OpenECAD 3.1B	✓	✓	×	38.59	40.69	38.75	-
GPT-4o-mini	×	✓	✓	56.28	56.12	30.54	0.4462
GPT-4o-mini	✓	✓	✓	60.54	65.94	53.01	0.5185
Gemini-1.5 Flash	×	✓	✓	64.24	63.81	52.24	0.4531
Gemini-1.5 Flash	✓	✓	✓	64.15	67.62	54.67	0.4618
Gemini-2.0 Flash (no sample)	×	✓	✓	13.82	13.72	10.61	-
Gemini-2.0 Flash	×	✓	✓	65.07	71.09	56.30	0.3899
Gemini-2.0 Flash	✓	✓	✓	**68.07**	**72.22**	**59.26**	**0.5753**

Table 4: Performance comparison of our model with and without RAG, across sketch and text modalities with OpenECAD [16]. The table includes curve accuracy, loop accuracy, and weighted average score (%) for test data.

7 REFERENCES

[1] Heidari, N. *et al.* Geometric deep learning for computer-aided design: A survey. *arXiv preprint arXiv:2402.17695* (2024). URL `https://arxiv.org/abs/2402.17695`.

[2] Su, H. *et al.* Multi-view convolutional neural networks for 3d shape recognition (2015). URL `https://arxiv.org/abs/1505.00880`. `1505.00880`.

[3] Hamdi, A. *et al.* Mvtn: Multi-view transformation network for 3d shape recognition (2021). URL `https://arxiv.org/abs/2011.13244`. `2011.13244`.

[4] Qi, C. R. *et al.* Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[5] Wang, Y. *et al.* Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics* (2019). URL `https://doi.org/10.1145/3326362`.

[6] Pan, X. *et al.* 3d shape retrieval by poisson histogram. *Pattern Recognition Letters* (2011). URL `https://www.sciencedirect.com/science/article/pii/S0167865511000055`.

[7] Yoon, S. M. *et al.* 3d model retrieval using the histogram of orientation of suggestive contours. In *Advances in Visual Computing* (2011).

[8] Wang, F. *et al.* Sketch-based 3d shape retrieval using convolutional neural networks (2015). URL `https://arxiv.org/abs/1504.03504`. `1504.03504`.

[9] Radford, A. *et al.* Learning transferable visual models from natural language supervision (2021). URL `https://arxiv.org/abs/2103.00020`. `2103.00020`.

[10] Guo, H. *et al.* Complexgen: Cad reconstruction by b-rep chain complex generation (2022). URL `https://arxiv.org/abs/2205.14573`. `2205.14573`.

[11] Guillard, B. *et al.* Sketch2mesh: Reconstructing and editing 3d shapes from sketches (2021). URL `https://arxiv.org/abs/2104.00482`. `2104.00482`.

[12] Cheng, Y.-C. *et al.* Sdfusion: Multimodal 3d shape completion, reconstruction, and generation (2023). URL `https://arxiv.org/abs/2212.04493`. `2212.04493`.

[13] Wu, R. *et al.* Deepcad: A deep generative network for computer-aided design models (2021). URL `https://arxiv.org/abs/2105.09492`. `2105.09492`.

[14] Uy, M. A. *et al.* Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders (2022). URL `https://arxiv.org/abs/2112.09329`. `2112.09329`.

[15] Li, C. *et al.* Free2cad: Parsing freehand drawings into cad commands. *ACM Transactions on Graphics* (2022). URL `https://doi.org/10.1145/3528223.3530133`.

[16] Yuan, Z. *et al.* Openecad: An efficient visual language model for editable 3d-cad design. *Computers & Graphics* (2024). URL `https://doi.org/10.1016/j.cag.2024.104048`.

[17] Khan, M. S. *et al.* Text2cad: Generating sequential cad models from beginner-to-expert level text prompts (2024). URL `https://arxiv.org/abs/2409.17106`. `2409.17106`.

[18] Lin, F. *et al.* Zero-shot everything sketch-based image retrieval, and in explainable style (2023). URL `https://arxiv.org/abs/2303.14348`. `2303.14348`.

[19] OpenAI *et al.* Gpt-4 technical report (2024). URL `https://arxiv.org/abs/2303.08774`. `2303.08774`.

[20] Lewis, P. *et al.* Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)* (2020).

[21] Koch, S. *et al.* Abc: A big cad model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[22] Xu, X. *et al.* Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks (2022). URL `https://arxiv.org/abs/2207.04632`. `2207.04632`.

[23] Xu, X. *et al.* Hierarchical neural coding for controllable cad model generation. *arXiv preprint arXiv:2307.00149* (2023).

[24] Badagabettu, A. *et al.* Query2cad: Generating cad models using natural language queries (2024). URL `https://arxiv.org/abs/2406.00144`. `2406.00144`.

[25] Qwen *et al.* Qwen2.5 technical report (2025). URL `https://arxiv.org/abs/2412.15115`. `2412.15115`.

[26] Team, G. *et al.* Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context (2024). URL `https://arxiv.org/abs/2403.05530`. `2403.05530`.

[27] Bhunia, A. K. *et al.* Vectorization and rasterization: Self-supervised learning for sketch and handwriting (2021). URL `https://arxiv.org/abs/2103.13716`. `2103.13716`.

[28] Lin, H. *et al.* Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt (2020). URL `https://arxiv.org/abs/2005.09159`. `2005.09159`.

[29] Manda, B. *et al.* CADSketchNet - an annotated sketch dataset for 3d cad model retrieval with deep neural networks. *Computers & Graphics* (2021). URL `https://doi.org/10.1016/j.cag.2021.07.001`.

[30] Kim, S. *et al.* A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)* (2020).

[31] Kosalaraman, K. K. *et al.* Sketchclean-gan: A generative network to enhance and correct query sketches for improving 3d cad model retrieval systems. *Computers & Graphics* (2024). URL `https://www.sciencedirect.com/science/article/pii/S0097849324001353`.

[32] Kendre, P. P. *et al.* Sketchcadgan: A generative approach for completing partially drawn query sketches of engineering shapes to enhance retrieval system performance. *Computers & Graphics* (2023). URL `https://www.sciencedirect.com/science/article/pii/S0097849323001243`.

[33] Tang, Y. *et al.* Minigpt-3d: Efficiently aligning 3d point clouds with large language models using 2d priors (2024). URL `https://arxiv.org/abs/2405.01413`. `2405.01413`.

[34] Xue, L. *et al.* Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding (2023). URL `https://arxiv.org/abs/2212.05171`. `2212.05171`.

[35] Wu, Z. *et al.* 3d shapenets: A deep representation for volumetric shapes (2015). URL `https://arxiv.org/abs/1406.5670`. `1406.5670`.

[36] Fang, Z. *et al.* Modelnet-o: A large-scale synthetic dataset for occlusion-aware point cloud classification (2024). URL `https://arxiv.org/abs/2401.08210`. `2401.08210`.

[37] Li, M. *et al.* Photo-sketching: Inferring contour drawings from images (2019). URL `https://arxiv.org/abs/1901.00542`. `1901.00542`.

[38] Simo-Serra, E. *et al.* Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics* (2016). URL `https://doi.org/10.1145/2897824.2925972`.

[39] Li, X. *et al.* 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* (2024). URL `https://doi.org/10.1145/3664647.3681549`.

[40] Jiang, A. Q. *et al.* Mistral 7b (2023). URL `https://arxiv.org/abs/2310.06825`. `2310.06825`.

[41] Yu, J. *et al.* Coca: Contrastive captioners are image-text foundation models (2022). URL `https://arxiv.org/abs/2205.01917`. `2205.01917`.

[42] Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale (2021). URL `https://arxiv.org/abs/2010.11929`. `2010.11929`.

[43] Devlin, J. *et al.* Bert: Pre-training of deep bidirectional transformers for language understanding (2019). URL `https://arxiv.org/abs/1810.04805`. `1810.04805`.

[44] Zhou, B. *et al.* Tinyllava: A framework of small-scale large multimodal models (2024). URL `https://arxiv.org/abs/2402.14289`. `2402.14289`.

[45] Team, G. *et al.* Gemma: Open models based on gemini research and technology (2024). URL `https://arxiv.org/abs/2403.08295`. `2403.08295`.