QueryCAD: Grounded Question Answering for CAD Models

Claudius Kienle[1], Benjamin Alt[1], Darko Katic[1] and Rainer Jäkel[1]

Abstract—CAD models are widely used in industry and are essential for robotic automation processes. However, these models are rarely considered in novel AI-based approaches, such as the automatic synthesis of robot programs, as there are no readily available methods that would allow CAD models to be incorporated for the analysis, interpretation, or extraction of information. To address these limitations, we propose QueryCAD, the first system designed for CAD question answering, enabling the extraction of precise information from CAD models using natural language queries. QueryCAD incorporates SegCAD, an open-vocabulary instance segmentation model we developed to identify and select specific parts of the CAD model based on part descriptions. We further propose a CAD question answering benchmark to evaluate QueryCAD and establish a foundation for future research. Lastly, we integrate QueryCAD within an automatic robot program synthesis framework, validating its ability to enhance deep-learning solutions for robotics by enabling them to process CAD models. **https://claudius-kienle.github.io/querycad**

I. INTRODUCTION

In the industrial sector, many workflows are centered around Computer-Aided Design (CAD) models. These models contain precise representations of individual parts and their assembly into larger components. Engineers in various industrial domains, such as robot programming, rely heavily on CAD models to extract measurements, identify specific features, and understand how parts interact within a larger system. To automate these still largely manual engineering processes, it is crucial to develop methods that can retrieve and interpret information from CAD models in an automated manner, mirroring the way users interact with them. However, since no web-scale dataset for industrial CAD models exists, it is generally hard to train or extend existing models to understand this modality [1]. Moreover, there currently exists no method for automatic information retrieval from CAD models. This gap is particularly evident in the context of automated robot program synthesis for industrial applications, where extracting parameters from CAD models is essential.

In response to this, we propose QueryCAD, the first deep learning-based question-answering system for CAD models. QueryCAD is the first system to retrieve specific information like measurements, positions, or counts of features or parts of CAD models in response to natural-language questions. Given a free-text question, QueryCAD generates answers
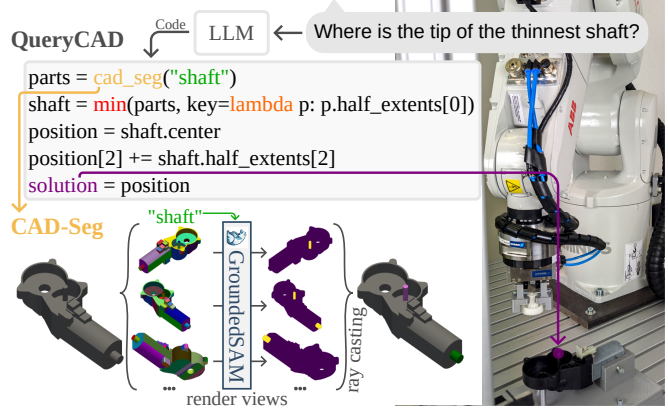
Fig. 1. QueryCAD computes precise measurements in response to natural-language queries. It leverages a code-writing LLM with in-context prompting to generate executable code, which then retrieves specific parts of the CAD model through multi-view segmentation. This process allows for extraction of detailed measurements from CAD models based on the high-level user queries. The computed measurements can be utilized in downstream applications such as automatic robot program parameterization, CNC machining or 3D printing.

by grounding its reasoning process in structured engineering knowledge, directly retrieving measurements from the CAD model. Grounding natural-language queries in CAD data enables seamless integration of CAD models into deep learning-based applications and is particularly compatible with existing frameworks that build upon Large Language Models (LLMs) [2], [3]. QueryCAD is integrated into the MetaWizard robot program synthesis system [3] to enable the automatic generation of industrial robot programs given natural-language task specifications.

In this paper, we make the following contributions:

1) **SegCAD**: A model for open-vocabulary multi-view CAD part segmentation.
2) **QueryCAD**: The first system for CAD question answering.
3) **CAD-Q&A Benchmark**: A benchmark for CAD question answering, containing 111 questions about 18 different CAD models.
4) **Robot Program Synthesis**: Integration of QueryCAD in a robot program synthesis framework for automatic generation of robot programs grounded in CAD data.

II. RELATED WORK

A. 3D Part Segmentation

3D part segmentation focuses on methods for partitioning 3D objects represented as voxels, meshes, or point clouds into their functional components, e.g. separating a hammer

arXiv:2409.08704v2 [cs.RO] 16 Sep 2024

into its handle and head. These methods can be categorized based on their input modalities, whether they utilize prompts for segmentation, as well as the type of prompts used.

Qi et al. [4], [5] introduced PointNet, a neural network designed to segment point clouds into coherent parts and trained on extensive part segmentation datasets. Subsequent work by Qian et al. [6] enhance PointNet's performance by optimizing its training procedure and scaling strategies.

To specify the parts to be segmented, some methods leverage text-based prompts to guide part segmentation. Liu et al. [7], [8] utilize the pretrained image-language model GLIP [9], which benefits from its extensive training on web-scale images. They employ a multi-view detection strategy to convert 2D bounding boxes into 3D segmentations, enhancing zero-shot generalization and open-set detection. The language modality of GLIP enables specification of segmented parts using text-based descriptions.

Another recent advancement involves using point-based prompts for segmentation. Zhou et al. [10] introduce a 3D segmentation model capable of segmenting parts based on a point provided as prompt. This approach allows to specify the part to segment by indicating a point on the object.

There is a wide range of methods for part segmentation, each utilizing different representations and techniques. Many approaches focus on point clouds [4], [5], [7], [8], [10], while some also leverage voxel-based representations [11]. However, no current methods directly segment CAD models into parts. Furthermore, existing approaches do not segment parts based on text-based descriptions. Recent methods predict bounding boxes [7], [8], which is too imprecise for CAD models where more detailed retrieval via segmentation masks is needed.

B. 3D Scene Understanding

In research, there is growing interest in solving tasks related to 3D scenes. Most prominently, 3D question answering revolves around answering natural-language questions about objects in a scene [12]–[17]. Huang et al. [12] propose an LLM-based model that works with object identifiers of a 3D scene and can answer questions about objects as well as identify the object asked about in the question. 3D visual grounding [12], [16], [18]–[22] is the task to locate a object in a 3D scene by natural language. Zhao et al. [18] propose a model that processes the raw point cloud and fuses it with the text to predict a bounding box that surrounds the target object. The finetuned LLM proposed by Huang et al. [12] was also applied for visual grounding by determining the object identifier that matches with the natural language description of the target object.

There is a considerable body of work on 3D (indoor) scene question answering and integrating natural language with point-cloud represented scenes. However, to the best of our knowledge, no work exists on transferring these approaches for question answering tasks on CAD models or parts.

C. CAD Machining Feature Classification

Numerous approaches have been developed for the automatic classification of operations required for machining a CAD model [1], [23]–[33]. These methods aim to assign specific machining features to each face of the CAD model from a predefined set of features. The approaches can be categorized based on the modality the classification model operates on.

One common strategy is to represent the CAD model as a graph, with two prevalent types of graph representations found in the literature. In a *BRep-Graph*, faces are represented as nodes, where two nodes are interconnected if they are adjacent on the CAD model [23]–[26]. A *Facet-Graph* represents the facets of the triangulated mesh of the CAD model [1], [27]–[29]. The node and edge features vary across different approaches but consistently include properties of the face or facet. To classify the nodes, these approaches employ Graph Neural Networks (GNNs). Notably, Colligan et al. [1] combine a *BRep-Graph* and a *Facet-Graph* into a hierarchical GNN to classify each node according to its machining feature.

In another set of approaches, the CAD model is sampled to a point cloud [30] or voxel grid [31]–[33]. These methods then train a 3D segmentation network to classify each point or voxel by its corresponding machining feature. For example, Lei et al. [30] sample a point cloud from the CAD model and train a 3D segmentation network to classify each point into one of 33 machining features.

All approaches share the common requirement of training a neural network for the classification task. This necessitates a large dataset, which is often generated synthetically [1], [23], [31] because existing large-scale real-world datasets [34], [35] lack the necessary labels [1]. While these methods are highly effective for machining classification and show a good generalization to new shapes, they are limited by their closed-set classification, making it difficult to adapt them for identifying additional feature types. Furthermore, these approaches typically operate at a low feature level, focusing on individual features rather than classifying higher-level features such as parts or more complex structures.

III. METHODS

We propose QueryCAD, a system to answer free-form questions about CAD models and their parts. In doing so, QueryCAD accesses SegCAD, a model for segmenting parts of a CAD model matching an open-vocabulary part description. The architecture can be seen in Figure 2. To evaluate QueryCAD, we develop the first benchmark for the task of CAD question answering.

A. SegCAD: Open-Set CAD Segmentation

In CAD question answering, the goal is to retrieve specific information about parts or individual features of a CAD model. This process begins with identifying the parts or features referenced in a given question. To achieve this, we developed SegCAD, a CAD instance segmentation model specifically designed to identify parts or features based on free-text part descriptions.

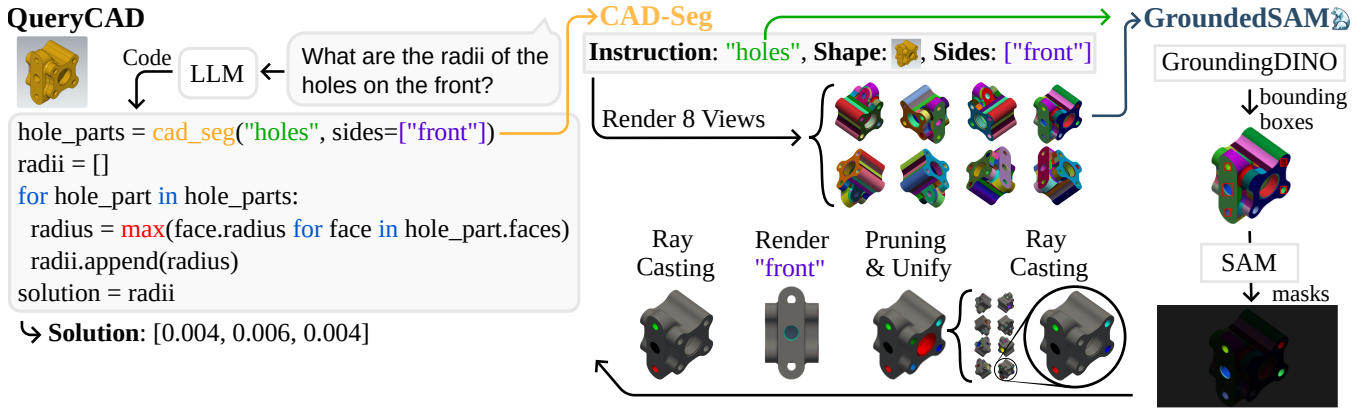The CAD model is first *3D rendered* (III-A.1) from predefined viewing angles, with each rendered 2D image

QueryCAD

Code → LLM ← What are the radii of the holes on the front?

```python
hole_parts = cad_seg("holes", sides=["front"])
radii = []
for hole_part in hole_parts:
    radius = max(face.radius for face in hole_part.faces)
    radii.append(radius)
solution = radii
```

↳ **Solution**: [0.004, 0.006, 0.004]

CAD-Seg

Instruction: "holes", **Shape**: 🔧, **Sides**: ["front"]

Render 8 Views

Ray Casting | Render "front" | Pruning & Unify | Ray Casting

GroundedSAM

GroundingDINO

↓ bounding boxes

SAM

↓ masks

Fig. 2. QueryCAD: Given a free-form question, the LLM generates Python code to compute the required measurements or properties of the CAD model. The generated code invokes SegCAD to identify the relevant parts of the CAD model specified in the question. SegCAD renders the CAD model from multiple views and performs ray casting to align the rendered images with the CAD faces. Parts that match the description, such as "holes", are retrieved and returned as Python objects for further processing. The final answer is computed by the generated Python code and stored as variable *solution*.

showing the CAD model from a unique perspective. These images are then processed through a promptable *instance segmentation* model [36] (III-A.2), which masks the parts in the image that correspond to the provided part description. Finally, the masked image is unprojected back into 3D space via *image to CAD model alignment* (III-A.3), allowing us to identify the CAD faces that were masked by the instance segmentation model.

1) 3D Rendering: Many CAD models are colored in one color, which makes it hard for the human eye and especially harder for a segmentation model to accurately identify the shape and features of the model. Therefore, given a defined viewing angle, we paint every face of the model visible from the viewing angle randomly in distinct colors and render it via orthographic projection [37]. This ensures that we only see the faces visible from the side we render the model from and not see faces from other sides as it would happen with perspective projection. Lastly, we take a 1920x1080 pixel image of the rendered model. The high resolution of the image ensures capturing small features, like small holes or protrusions, in good detail. Images with a lower resolution make it harder for the segmentation model to detect all features accurately.

2) Instance Segmentation: To segment the rendered CAD model based on a free-text description, we utilize GroundingDINO [38] and Segment Anything Model (SAM) [39]. These models were selected due to the diversity of their training datasets and their demonstrated generalization capabilities across various domains [40]–[42]. In contrast, alternative approaches like Point-LLM [10] were trained on smaller, more specialized datasets, which limits their generalization potential. Additionally, our segmentation approach facilitates answering view-related questions by allowing us to render the model from specific angles.

GroundingDINO processes the rendered CAD image along with the free-text part description and outputs multiple bounding boxes with associated probabilities. We determined that a probability threshold of 30 % was optimal (see

Ablation IV-B). Bounding boxes with probabilities below this threshold are discarded. Finally, we apply SAM [39] to each bounding box and rendered CAD image to generate precise segmentation masks. This combination of GroundingDINO and SAM was first proposed by Ren et al. [36].

GroundingDINO tends to select large portions or even the entire CAD model as a single bounding box. To address this, we filter out masks that cover more than 45 % of the rendered CAD model. The segmentation process can be seen in Figure 2.

3) Image to CAD Model Alignment: The 2D segmentation masks identify the pixels in the rendered view that correspond to parts matching the part description. To determine the faces of the CAD model displayed at the masked pixels, we use ray casting [43]. However, because the segmentation mask operates on a 2D rendering of the CAD model and has no pixel-level segmentation accuracy, additional post-processing is necessary to refine the selection of faces.

First, segmentation masks often cover small portions of neighboring faces. To filter these out, we only select a face if more than 5 % of its visible surface in the rendered image is covered by the segmentation mask.

Second, adjacent regions in the rendered CAD image do not always correspond to adjacent faces on the CAD model itself, such as when there is content behind a hole. The segmentation mask generated by SAM sometimes selects faces that are adjacent in the 2D image but not on the actual CAD model. We require that a segmentation mask includes only faces that are truly adjacent in the 3D space. To enforce this, we perform a post-pruning of the CAD faces, depicted in Figure 3. Using ray casting, we calculate the distance of each face from the camera's viewport. Starting with the masked face closest to the viewport, we recursively select only those masked faces that are adjacent to it. This ensures that no faces are selected that are further back on the CAD model and not adjacent to the closer faces.

4) Multi-View Rendering and View-Specific Retrieval: We render the CAD model from multiple views since not all features are visible from a single viewing angle. We found

Face Pruning →

(1) 2D Mask

(3) Depth Ray Casting
■ Closest ■ Farthest

(5) Pruned Part

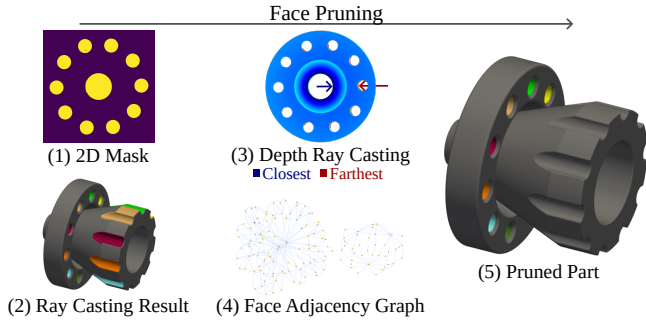(2) Ray Casting Result

(4) Face Adjacency Graph

Fig. 3. Pruning CAD faces to ensure only neighboring faces are selected as part. Starting with raw faces (2) derived from a 2D segmentation mask (1), we use ray casting to identify the face closest to the viewport (3). From this face, we select adjacent faces based on the adjacency graph (4) recursively. As a result, non-adjacent faces, particularly those further in the background, are discarded (5).

that 6 viewing angles yielded the best results on the CAD-Q&A Benchmark (see Ablation IV-B).

To address view-related questions, such as "How many protrusions are visible on the right?", SegCAD is designed to retrieve only parts or features visible from specific viewing directions. After identifying all parts or features that globally match the part description, we render the CAD model from the requested viewing angles (a subset of top, bottom, right, left, front, and back). Using the ray casting method described in III-A.3, we determine which parts are actually visible from these angles and discard those that are not.

Additionally, due to the orthographic projection, we do not render the CAD model exactly along the main axes, as many faces would be orthogonal to the viewport and thus not visible in the rendered image. Instead, we slightly perturb the viewing angle by one degree along the azimuth when rendering along the main axes.

B. QueryCAD: CAD Understanding, Reasoning and Answering

QueryCAD is the first system capable of answering questions about the features or parts of CAD models. A key focus of this system is its ability to provide precise measurements, positions, and other data by enabling LLMs to interact directly with CAD models. QueryCAD can accurately retrieve measurements for specific parts of the CAD model, such as dimensions, radii, center positions, and depths of features. These measurements can be used to filter relevant parts or to form a response.

The system is designed to handle a wide range of free-text questions, whether posed by other deep-learning systems or human engineers, without imposing any constraints on the types of objects or the structure of the questions.

To achieve these properties, we implemented the following approach: First, the user's query is passed to a code-writing LLM using a carefully crafted prompt[1]. The objective of the LLM is to generate Python code that computes the measurements that answer the question. For this, the prompt

[1]See https://claudius-kienle.github.io/querycad for prompts, CAD-Q&A benchmark, and examples.

defines Python classes for the CAD model and the CAD part. The Python classes have attributes such as extents or the center position, which can be accessed in the code generated by the LLM. To enable the LLM to search in the CAD model for a specific feature or part, SegCAD is integrated via a Python Application Programming Interface (API). The API calls are parameterized by the LLM with the free-text part description and potential viewing angles.

The LLM processes the prompt and generates Python code using Chain-of-Thought (CoT) prompting [44]. This technique enhances model transparency by having the model explain each reasoning step. We observed that CoT prompting considerably increases the share of correct model responses. The prompt contains 3 high-quality in-context samples.

We observed that constructing a prompt that clearly states the task without any ambiguities is difficult, but highly important to reliably answer user queries with the code written by the LLM. Especially when reasoning on 3D shapes, these ambiguities are not easy to spot but important to clarify. One example are the *extents* of an object, which can be defined as the full length of the object along global coordinate axes or the distance from the center of the object to its edge. Properties like the *width*, *height*, or *depth* of an object are similarly ambiguous. We curated a prompt to clarify these linguistic ambiguities, which are often featured in user queries. For example, the prompt clarifies to always use full extents unless otherwise stated via *half-extents* and analogously *half-{width, depth, height}*. Moreover, the prompt pushes the LLM to do any reasoning and conversion of metrics in Python code instead of implicitly converting the metrics, which we found to improve the performance and reduce hallucinations, especially for smaller models like Llama 3.1 8B [45]. This is especially important for conversions between units, such as converting meters to millimeters, which the model is prompted to do directly in Python.

The Python code generated by the LLM is executed, which in turn calls the SegCAD model if stated in the code. Examples of LLM predictions in response to user questions are shown in Figures 1 and 2. During execution, the code accesses the properties of the CAD models' parts (e.g. the radius of a hole or the center of a rod). This enables QueryCAD to filter and retrieve the relevant properties based on the user query. Finally, the result is returned as a response to the query.

C. CAD-Q&A Benchmark

At the time of writing, no dataset or benchmark for CAD question answering exists. To evaluate our approach and establish a common benchmark for future research, we have developed a highly curated and manually annotated dataset specifically tailored for CAD question answering, available on the paper's website.

Our CAD Q&A benchmark consists of 18 CAD models, 10 of which are derived from the ABC dataset [34]. We specifically selected models from this dataset that are complex and representative of real-world industrial applications.
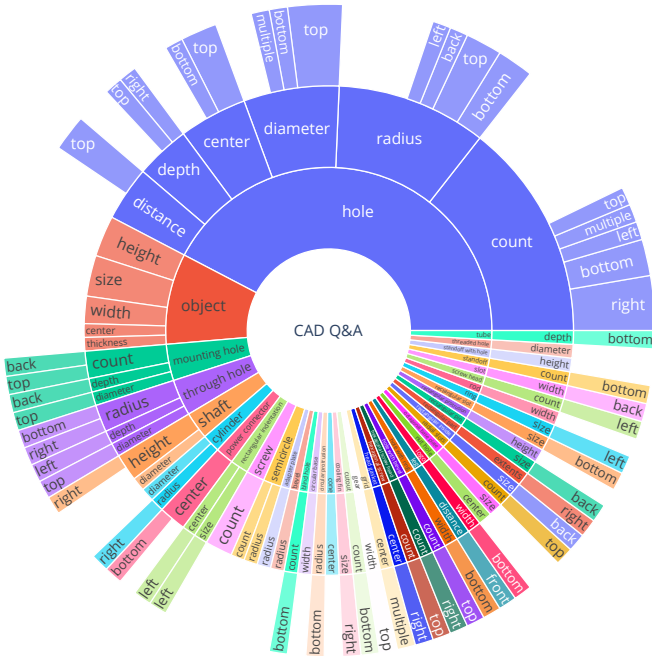
Fig. 4. Distribution of dataset questions. Every question is composed of a part it asks about (inner circle), a property to retrieve (middle circle) and optionally one or multiple sides the part must be visible from (outer circle). Some questions enforce additional filtering based on the part properties, like retrieving only parts with a radius of 5 mm.

	Llama 3 13B	Llama 3.1 8B	Llama 3.1 405B	GPT4o Aug-24
Correct	32	36	43	**44**
Wrong	79	75	68	67
Syntax	15	6	0	0
Reasoning	17	8	2	1
Masks	40	51	55	52
CAD-Interface	7	10	11	14

their parts or features. We use GPT4o [46] as LLM, but also compare it to open-source LLMs in Section IV-B.

Out of 111 questions, QueryCAD correctly answered 44, provided partially correct answers where the answer overlaps with the solution for 15 questions, and gave incorrect responses to the remaining 52. These results demonstrate that QueryCAD can answer a range of questions about CAD models. To more accurately understand how QueryCAD fails to answer the 67 wrongly answered samples, we categorize the types of mistakes it made into four groups, detailed in Table I.

Syntax addresses samples where the LLM generated invalid Python syntax, which was the case for none of the samples. *Reasoning* corresponds to samples where the proposed Python code features incorrect reasoning or hallucinations, which occurred once. If the Python code is valid, there are still two issues that can cause invalid answers: *Masks*, where incorrect parts were returned by SegCAD due to imprecise, missing or incorrect masks proposed by GroundedSAM or due to inaccuracies in the ray casting, which occurred 52 times. Lastly, *CAD-Interface* classifies answers where the CAD interface provided in the Python code was not powerful enough to answer the question, which was the case for 14 samples.

B. Ablations

a) LLM Backend: To assess the robustness of QueryCAD across different LLMs, we evaluated it using various models on the CAD-Q&A benchmark, as shown in Table I. Larger models like GPT4o [46] and Llama 3.1 405B [45] exhibit fewer syntax issues and demonstrate stronger reasoning abilities, leading to 8 more correctly answered questions. There is no significant difference between the predicted answers of Llama 3.1 405B and GPT4o, which shows that QueryCAD can be used effectively with open source models. Although the large models notably outperform the smaller Llama variants, it is noteworthy that Llama 3.1 8B [45], the smallest available version of Llama 3.1, still shows good reasoning on most samples, resulting in a modest increase of reasoning and syntax errors. This demonstrates that QueryCAD remains effective even when using lightweight and ultra-fast LLMs like Llama 3.1 8B, making it viable for deployment on edge devices.

b) Viewing Angles: We evaluated SegCAD with two different viewing angle configurations: 6 viewing angles along the main axes and 8 viewing angles from all 8 corners

---

Additionally, we incorporated 8 CAD models from real-world industrial settings to further enhance the benchmark's relevance.

For each CAD model, we generated between 4 and 10 questions designed to retrieve specific properties, such as measurements, positions, or counts of particular parts. Initially, we attempted to use LLMs like GPT4o to generate the questions, providing them with screenshots of the CAD models and a task description. However, the questions generated were often unclear and ambiguous. Therefore, we opted to handcraft the questions for each CAD model. These questions were then validated by an industrial engineer.

Each question targets either a specific type of part or the entire CAD model, inquiring about a particular property while sometimes restricting the valid parts by specifying a side the parts should be visible on or applying other filtering criteria based on the part's characteristics. The distribution and structure of these questions are illustrated in Figure 4. To address open-set segmentation, the questions cover a total of 43 diverse parts and ask about 11 different properties.

To define the label for each question, we manually measured the CAD models using a CAD kernel and calculated the responses by hand. The final dataset comprises 111 questions across 18 CAD models.

IV. RESULTS

A. Evaluation

We evaluate QueryCAD on the newly generated CAD Q&A Benchmark to quantitatively assess its ability to answer questions about the precise measurements of CAD models,
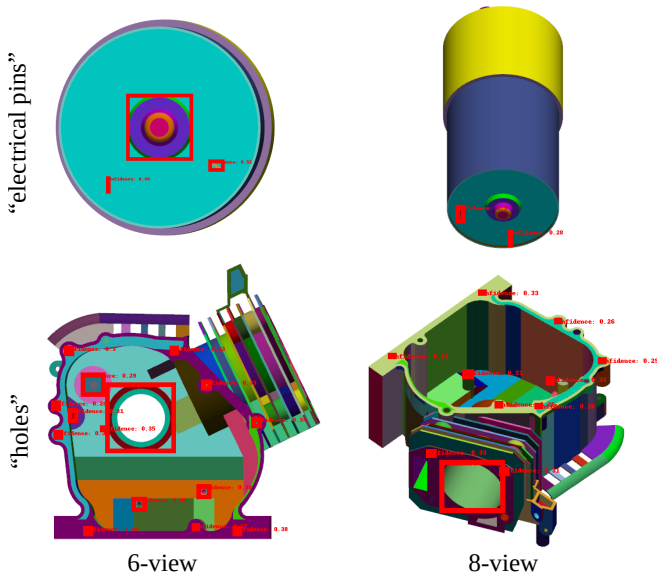
Fig. 5. We evaluate SegCAD by rendering the CAD model from either 6 views (left) or 8 views (right). GroundingDINO detects more parts when rendered along the main axes (lower left), but it occasionally selects parts that do not match the description (upper left). In contrast, rendering the model from 8 viewing angles often results in some parts not being selected (lower right) but reduces the likelihood of selecting incorrect parts (upper right).

of the CAD model with an azimuth and elevation of the camera of 45 degrees. Figure 5 compares the two viewing configurations. While the configuration with 6 viewing angles makes it harder for GroundedSAM [36] to understand the depth of the CAD model, as most of the features are orthogonal to the viewport, it effectively allows to detect features like through-holes or other geometrical features specifically from that axis. The configuration with 8 viewing angles has the benefit to capture the 3D shape of the model more accurately, reducing the likelihood of wrong predictions by GroundedSAM. One example are shafts and holes, which look the same from the top. However, we observed that with 8 viewing angles, GroundedSAM occasionally misses certain parts, as shown in Figure 5. Additionally, it becomes more challenging to select the parts visible from a specific viewing direction with 8 viewing angles.

On the CAD-Q&A dataset, using 6 viewing angles resulted in 44 correctly answered questions, while QueryCAD answered 28 answers correctly with 8 viewing angles. This mainly stems from the fact that SegCAD with 8 viewing angles often did not detect all parts the questions asked about or returned parts that were not visible from the sides considered in the question.

c) GroundingDINO Threshold: In SegCAD, the likelihood threshold for GroundingDINO's bounding box suggestions is a critical hyperparameter. Bounding boxes with a likelihood below this threshold are discarded. The threshold manages the trade-off between detecting parts with low scores, such as those that are only partially visible or very small, and the risk of selecting incorrect parts when the threshold is too low. We found that a threshold of 30 %

produced the best results on the CAD-Q&A benchmark with 44 correct answers compared to 41 correct answers with a threshold of 25 %.

C. Validation

QueryCAD can be integrated in existing deep-learning based methods in robotics to retrieve information directly from CAD models given a free-form question. We validate this by combining QueryCAD with the MetaWizard program synthesis framework [3]. During program synthesis, MetaWizard asks the user to define parameters of the robot program that cannot be derived automatically from its internal knowledge base. The integration of QueryCAD permits MetaWizard to directly ask QueryCAD, instead of the human user, in natural language about the parameters it needs.

MetaWizard with QueryCAD is employed to program an assembly task where the robot picks up a gear and inserts it into an engine block, as shown in Figure 1. The user can program the robot for this task in natural language. During the program synthesis, MetaWizard queries QueryCAD twice: first, to determine the center of the gear by asking "What is the center of the object?" on the gear's CAD model, and second, to locate the tip of the shaft where the gear must be inserted by asking "Where is the tip of the shaft?" on the engine block's CAD model. For details of the interaction and results, we refer to the paper's website.

V. CONCLUSION AND OUTLOOK

We introduce QueryCAD, the first system for question answering tasks on CAD models. Our method accurately identifies the parts of the CAD model referenced in a question and retrieves precise measurements by grounding its predictions in a CAD model. We demonstrate the effectiveness of our approach in handling open-vocabulary queries related to CAD models. To assess its performance, we develop and publish the first CAD-Q&A benchmark, a tool that future research can utilize for comparative evaluation. Furthermore, by integrating our system into an existing robot program synthesis method, we validate that it successfully provides automatic querying of CAD information to existing algorithms.

While QueryCAD performs well on the CAD-Q&A benchmark, there is still room for improvement. The robustness of our approach largely depends on the accuracy of part segmentation of the CAD model with SegCAD. The most common errors during question answering arise from incorrect or missed part detection and artifacts caused by ray casting. Enhancing the instance segmentation of SegCAD remains an open area for future research. Additionally, while QueryCAD can handle a wide range of queries, such as measurements, e.g. radius, diameter, width, and part counts, it currently calculates all metrics in world coordinates. This can lead to incorrect answers when questions pertain to a part's local orientation. Moreover, properties like surface normals are not yet supported, presenting another avenue for future research.

REFERENCES

[1] A. R. Colligan, T. T. Robinson, D. C. Nolan, Y. Hua, and W. Cao, "Hierarchical CADNet: Learning from B-Reps for Machining Feature Recognition," *Computer-Aided Design*, vol. 147, p. 103226, June 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448522000240

[2] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, *et al.*, "Code as Policies: Language Model Programs for Embodied Control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 9493–9500. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10160591?casa_token=WFKbBaJAs6IAAAAA:KPr_ULH6fkAWMMyLaS01pZ2_xkQEajIgZyrD6wkN1jKE-wfvtX3DOwk8Gmb26BqUCzNuS_gLgQ

[3] B. Alt, F. Stöckl, S. Müller, C. Braun, J. Raible, S. Alhasan, *et al.*, "RoboGrind: Intuitive and Interactive Surface Treatment with Industrial Robots," Feb. 2024, arXiv:2402.16542 [cs]. [Online]. Available: http://arxiv.org/abs/2402.16542

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Qi_PointNet_Deep_Learning_CVPR_2017_paper.html

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html

[6] G. Qian, Y. Li, H. Peng, and J. Mai, "PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies," *Advances in neural information processing systems*, vol. 35, pp. 23192–23204, 2022.

[7] M. Liu, Y. Zhu, H. Cai, S. Han, Z. Ling, F. Porikli, and H. Su, "PartSLIP: Low-Shot Part Segmentation for 3D Point Clouds via Pretrained Image-Language Models," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Vancouver, BC, Canada: IEEE, June 2023, pp. 21736–21746. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023/html/Liu_PartSLIP_Low-Shot_Part_Segmentation_for_3D_Point_Clouds_via_Pretrained_CVPR_2023_paper.html

[8] Y. Zhou, J. Gu, X. Li, M. Liu, Y. Fang, and H. Su, "PartSLIP++: Enhancing Low-Shot 3D Part Segmentation via Multi-View Instance Segmentation and Maximum Likelihood Estimation," Dec. 2023, arXiv:2312.03015 [cs]. [Online]. Available: http://arxiv.org/abs/2312.03015

[9] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, *et al.*, "Grounded Language-Image Pre-Training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10965–10975. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Li_Grounded_Language-Image_Pre-Training_CVPR_2022_paper.html?ref=blog.roboflow.com

[10] Y. Zhou, J. Gu, T. Y. Chiang, F. Xiang, and H. Su, "Point-SAM: Promptable 3D Segmentation Model for Point Clouds," June 2024, arXiv:2406.17741 [cs]. [Online]. Available: http://arxiv.org/abs/2406.17741

[11] B. Graham, M. Engelcke, and L. V. D. Maaten, "3D Semantic Segmentation with Submanifold Sparse Convolutional Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA: IEEE, June 2018, pp. 9224–9232. [Online]. Available: https://ieeexplore.ieee.org/document/8579059/

[12] H. Huang, Z. Wang, R. Huang, L. Liu, X. Cheng, Y. Zhao, *et al.*, "Chat-3D v2: Bridging 3D Scene and Large Language Models with Object Identifiers," Dec. 2023, arXiv:2312.08168 [cs]. [Online]. Available: http://arxiv.org/abs/2312.08168

[13] Z. Zhu, X. Ma, Y. Chen, Z. Deng, S. Huang, and Q. Li, "3D-VisTA: Pre-trained Transformer for 3D Vision and Text Alignment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 2911–2921. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Zhu_3D-VisTA_Pre-trained_Transformer_for_3D_Vision_and_Text_Alignment_ICCV_2023_paper.html

[14] M. Parelli, A. Delitzas, N. Hars, G. Vlassis, S. Anagnostidis, G. Bachmann, and T. Hofmann, "CLIP-Guided Vision-Language Pre-Training for Question Answering in 3D Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2023, pp. 5607–5612. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2023W/O-DRUM/html/Parelli_CLIP-Guided_Vision-Language_Pre-Training_for_Question_Answering_in_3D_Scenes_CVPRW_2023_paper.html

[15] D. Azuma, T. Miyanishi, S. Kurita, and M. Kawanabe, "ScanQA: 3D Question Answering for Spatial Scene Understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 19129–19139. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Azuma_ScanQA_3D_Question_Answering_for_Spatial_Scene_Understanding_CVPR_2022_paper.html

[16] D. Z. Chen, A. X. Chang, and M. Nießner, "ScanRefer: 3D Object Localization in RGB-D Scans Using Natural Language," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 202–221.

[17] X. Ma, S. Yong, Z. Zheng, Q. Li, Y. Liang, S.-C. Zhu, and S. Huang, "SQA3D: Situated Question Answering in 3D Scenes," Apr. 2023, arXiv:2210.07474 [cs]. [Online]. Available: http://arxiv.org/abs/2210.07474

[18] Z. Wang, H. Huang, Y. Zhao, L. Li, X. Cheng, Y. Zhu, *et al.*, "3DRP-Net: 3D Relative Position-aware Network for 3D Visual Grounding," July 2023, arXiv:2307.13363 [cs]. [Online]. Available: http://arxiv.org/abs/2307.13363

[19] L. Zhao, D. Cai, L. Sheng, and D. Xu, "3DVG-Transformer: Relation Modeling for Visual Grounding on Point Clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 2928–2937. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/html/Zhao_3DVG-Transformer_Relation_Modeling_for_Visual_Grounding_on_Point_Clouds_ICCV_2021_paper.html?ref=https://githubhelp.com

[20] Z. Wang, H. Huang, Y. Zhao, L. Li, X. Cheng, Y. Zhu, *et al.*, "Distilling Coarse-to-Fine Semantic Matching Knowledge for Weakly Supervised 3D Visual Grounding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 2662–2671. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Wang_Distilling_Coarse-to-Fine_Semantic_Matching_Knowledge_for_Weakly_Supervised_3D_Visual_ICCV_2023_paper.html

[21] S. Chen, P.-L. Guhur, M. Tapaswi, C. Schmid, and I. Laptev, "Language Conditioned Spatial Relation Reasoning for 3D Object Grounding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20522–20535, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/819aaee144cb40e887a4aa9e781b1547-Abstract-Conference.html

[22] S. Huang, Y. Chen, J. Jia, and L. Wang, "Multi-View Transformer for 3D Visual Grounding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 15524–15533. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Huang_Multi-View_Transformer_for_3D_Visual_Grounding_CVPR_2022_paper.html

[23] W. Cao, T. Robinson, Y. Hua, F. Boussuge, A. R. Colligan, and W. Pan, "Graph Representation of 3D CAD Models for Machining Feature Recognition With Deep Learning," in *International design engineering technical conferences and computers and information in engineering conference*, vol. 84003. American Society of Mechanical Engineers Digital Collection, 2020, p. V11AT11A003. [Online]. Available: https://dx.doi.org/10.1115/DETC2020-22355

[24] M. H. Cha and B. C. Kim, "Machining feature recognition using BRepNet," *Journal of Mechanical Science and Technology*, vol. 37, no. 12, pp. 6103–6113, Dec. 2023. [Online]. Available: https://doi.org/10.1007/s12206-023-2403-4

[25] J. G. Lambourne, K. D. D. Willis, P. K. Jayaraman, A. Sanghi, P. Meltzer, and H. Shayani, "BRepNet: A Topological Message Passing System for Solid Models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12773–12782. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Lambourne_BRepNet_A_Topological_Message_Passing_System_for_Solid_Models_CVPR_2021_paper.html

[26] P. K. Jayaraman, A. Sanghi, J. G. Lambourne, K. D. D. Willis,

T. Davies, H. Shayani, and N. Morris, "UV-Net: Learning From Boundary Representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 11703–11712. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Jayaraman_UV-Net_Learning_From_Boundary_Representations_CVPR_2021_paper.html

[27] Y. Feng, Y. Feng, H. You, X. Zhao, and Y. Gao, "MeshNet: Mesh Neural Network for 3D Shape Representation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 8279–8286, July 2019, number: 01. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4840

[28] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: a network with an edge," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 90:1–90:12, July 2019. [Online]. Available: https://doi.org/10.1145/3306346.3322959

[29] J. Yang, K. Mo, Y.-K. Lai, L. J. Guibas, and L. Gao, "DSG-Net: Learning Disentangled Structure and Geometry for 3D Shape Generation," *ACM Trans. Graph.*, vol. 42, no. 1, pp. 1:1–1:17, Aug. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3526212

[30] R. Lei, H. Wu, and Y. Peng, "MFPointNet: A Point Cloud-Based Neural Network Using Selective Downsampling Layer for Machining Feature Recognition," *Machines*, vol. 10, no. 12, p. 1165, Dec. 2022, number: 12 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2075-1702/10/12/1165

[31] Z. Zhang, P. Jaiswal, and R. Rai, "FeatureNet: Machining feature recognition based on 3D Convolution Neural Network," *Computer-Aided Design*, vol. 101, pp. 12–22, Aug. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448518301349

[32] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1912–1920, iSSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/document/7298801

[33] D. Peddireddy, X. Fu, H. Wang, B. G. Joung, V. Aggarwal, J. W. Sutherland, and M. Byung-Guk Jun, "Deep Learning Based Approach for Identifying Conventional Machining Processes from CAD Data," *Procedia Manufacturing*, vol. 48, pp. 915–925, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920315821

[34] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, *et al.*, "ABC: A Big CAD Model Dataset for Geometric Deep Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9601–9611. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Koch_ABC_A_Big_CAD_Model_Dataset_for_Geometric_Deep_Learning_CVPR_2019_paper.html

[35] S. Kim, H.-G. Chi, X. Hu, Q. Huang, and K. Ramani, "A Large-Scale Annotated Mechanical Components Benchmark for Classification and Retrieval Tasks with Deep Neural Networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, Dec. 2020, pp. 175–191.

[36] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, *et al.*, "Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks," Jan. 2024, arXiv:2401.14159 [cs]. [Online]. Available: http://arxiv.org/abs/2401.14159

[37] P. Maynard, *Drawing Distinctions: The Varieties of Graphic Expression*. Cornell University Press, 2005, google-Books-ID: 4Y_YqOlXoxMC.

[38] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, *et al.*, "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," July 2024, arXiv:2303.05499 [cs]. [Online]. Available: http://arxiv.org/abs/2303.05499

[39] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, *et al.*, "Segment Anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 4015–4026. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Kirillov_Segment_Anything_ICCV_2023_paper.html

[40] "Papers with Code - COCO minival Benchmark (Object Detection)." [Online]. Available: https://paperswithcode.com/sota/object-detection-on-coco-minival?p=grounding-dino-marrying-dino-with-grounded

[41] "Papers with Code - ODinW Benchmark (Zero-Shot Object Detection)." [Online]. Available: https://paperswithcode.com/sota/zero-shot-object-detection-on-odinw?p=grounding-dino-marrying-dino-with-grounded

[42] "Papers with Code - MSCOCO Benchmark (Zero-Shot Object Detection)." [Online]. Available: https://paperswithcode.com/sota/zero-shot-object-detection-on-mscoco?p=grounding-dino-marrying-dino-with-grounded

[43] A. S. Glassner, *An Introduction to Ray Tracing*. Morgan Kaufmann, Jan. 1989, google-Books-ID: YPblYyLqBM4C.

[44] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[45] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, *et al.*, "The Llama 3 Herd of Models," Aug. 2024, arXiv:2407.21783 [cs]. [Online]. Available: http://arxiv.org/abs/2407.21783

[46] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, *et al.*, "GPT-4 Technical Report," Mar. 2024, arXiv:2303.08774 [cs]. [Online]. Available: http://arxiv.org/abs/2303.08774