Supplementary Materials for M3DBench

Mingsheng Li[1], Xin Chen[2], Chi Zhang[2], Sijin Chen[1], Hongyuan Zhu[3], Fukun Yin[1], Zhuoyuan Li[1], Gang Yu[2], and Tao Chen[1, ✉]

[1] School of Information Science and Technology, Fudan University, Shanghai, China
limc22@m.fudan.edu.cn, eetchen@fudan.edu.cn
[2] Tencent PCG, Shanghai, China
[3] Institute for Infocomm Research (I^{2}R) & Centre for Frontier AI Research (CFAR), A*STAR, Singapore

This supplementary material provides further details about M3DBench (Sec. 1), quantitative experiments (Sec. 2) on a broader range of tasks, additional analyses involving the scaling of model capacity and data size (Sec. 3), implementation details (Sec. 4) of baseline model and prompt for GPT-4 evaluation (Sec. 5).

1 Dataset

In Sec. 1.1, we provide more examples in M3Dbench for each task. Following that, we will introduce the dataset construction in Sec. 1.2 and provide statistics for the evaluation dataset in Sec. 1.3.

1.1 More Examples

1.2 Dataset Construction

In this work, we introduce a comprehensive 3D instruction tuning dataset, M3DBench, which serves as the foundation for developing versatile and practical general-purpose assistants in the real-world 3D environment. To construct M3DBench, we utilize 3D data from publicly available datasets [7, 3, 1, 5, 16, 8, 4] and propose a three-stage data generation pipeline. In the initial phase, we manually design templates and system messages for both template-based and LLM-prompting methods to create instruction-response pairs. From Tabs. 1 to 8, we provide a comprehensive overview of the prompts designed for various 3D tasks, each comprising task descriptions, specific requirements, and few-shot examples. For tasks such as object detection, we craft templates for instruction and response, then replace the template's keywords with annotations to construct instruction-response data [15], as illustrated in Tab. 9 and Tab. 10. Furthermore, we have developed an interleaved multi-modal instruction formula, where specific instance identifier *<target>* within the instructions are replaced with proposed multi-modal prompts, as illustrated in Tab. 11.

1.3 Evaluation Dataset

To quantitatively evaluate the effectiveness of instruction-tuned MLMs, we constructed an evaluation dataset to assess the models' performance across various

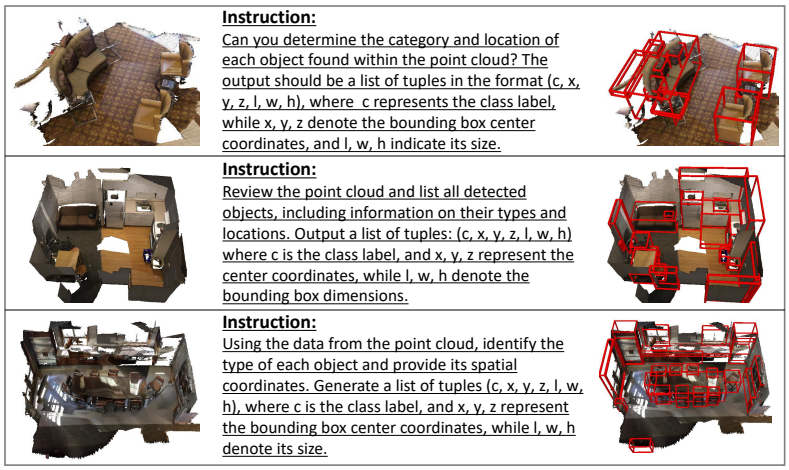


Fig. 1: Examples of 3D object detection. The left column represents the 3D scene, the middle column displays the instructions, and the right column shows the annotations for the object detection task. We save annotations in textual format and for visualization purposes here, we extract the bounding boxes from the text.

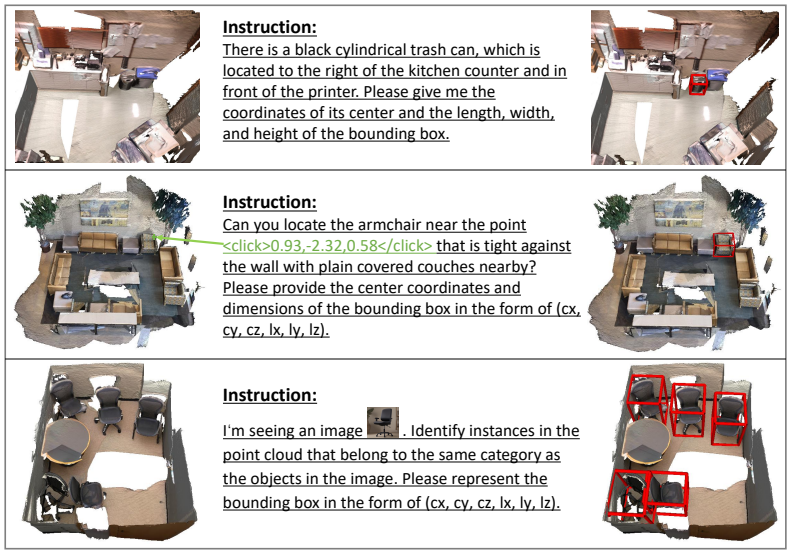


Fig. 2: Examples of 3D visual grounding. The left column represents the 3D scene, the middle column displays the instructions, and the right column shows the annotations for the visual grounding. M3DBench includes interleaved multi-modal instructions, and the annotations extend beyond annotating a single target object, encompassing the identification of multiple objects.

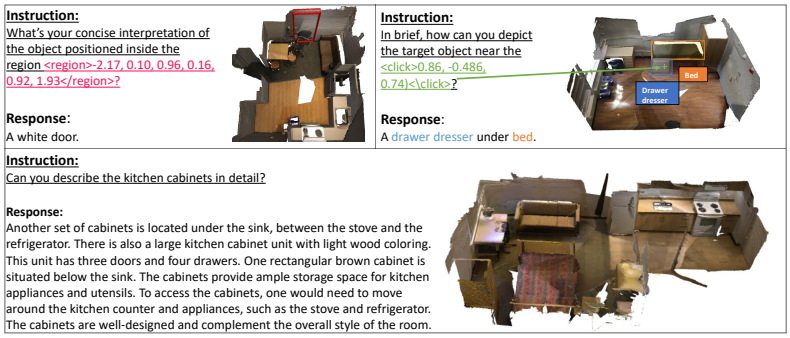


Fig. 3: Examples of 3D dense caption. We design diverse multi-modal instructions for dense captions for M3DBench. Additionally, we introduce terms such as *brief* or *detailed* within instructions to generate either concise titles or detailed descriptions for objects.

Fig. 4: Examples of 3D visual question answering. M3DBench comprises open-ended, free-form questions involving instance location, shape and size, object count, scene type, object and room functionality, and more. For instance, when asked about the functionality of a *rectangular silver object* in the upper-right scene, the answer begins by identifying the object and then describing its functionality. Furthermore, the two examples below illustrate instances where there might be multiple objects of *the same category* in the scene.

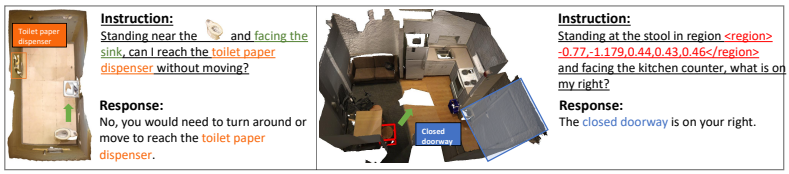


Fig. 5: Examples of embodied question answering. Embodied question answering requires the agent to understand the surrounding environment in order to answer questions under that situation.

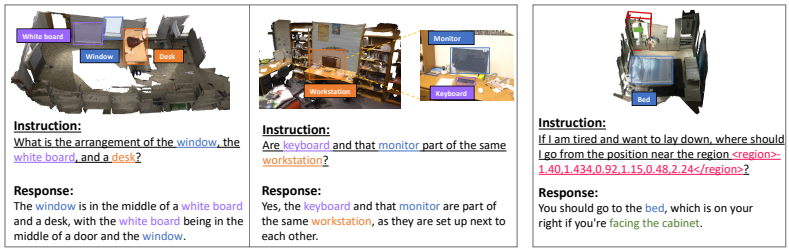


Fig. 6: Examples of multi-region reasoning (left) and embodied planning (right). In multi-region reasoning tasks (left), at least two objects are involved, querying their relative relationships and sizes, which enables a detailed comprehension of the scene. On the other hand, embodied planning (right) requires an agent to perceive the environment, understand the user's intentions, and then generate appropriate responses or actions to achieve predetermined goals.

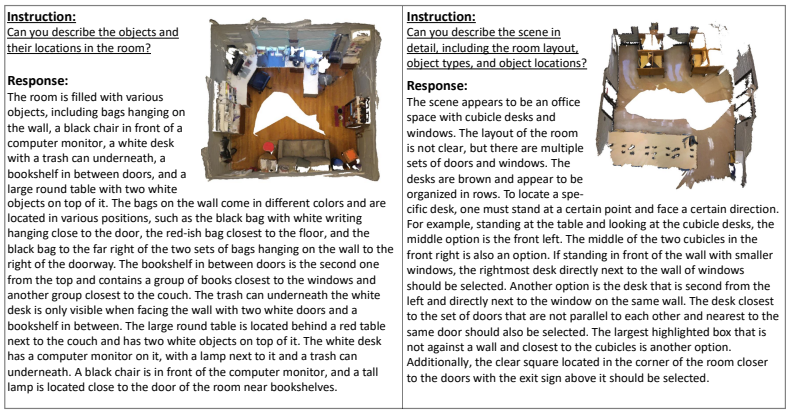


Fig. 7: Examples of scene description. Scene description requires integrating the visual information of the entire scene to generate a detailed and comprehensive introduction. This description will encompass aspects such as objects within the scene, the type of room, its functionality, and so on.

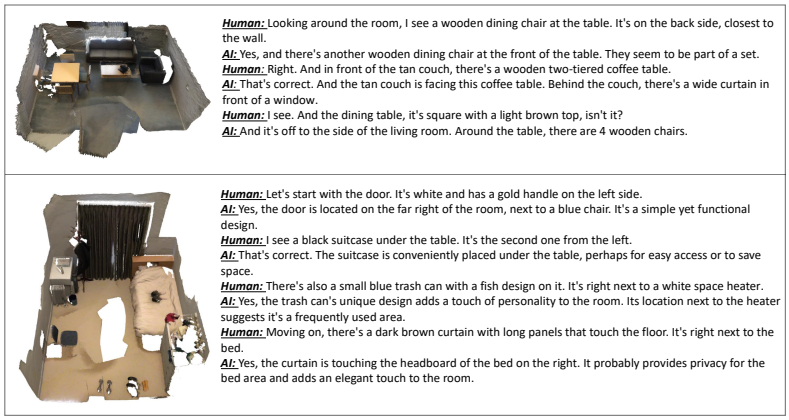


Fig. 8: Examples of multi-round dialogue. Multi-round dialogue necessitates the agent's ability to engage in natural and coherent communication with humans. This capability involves not only understanding and generating language but also ensuring accuracy and coherence in context.

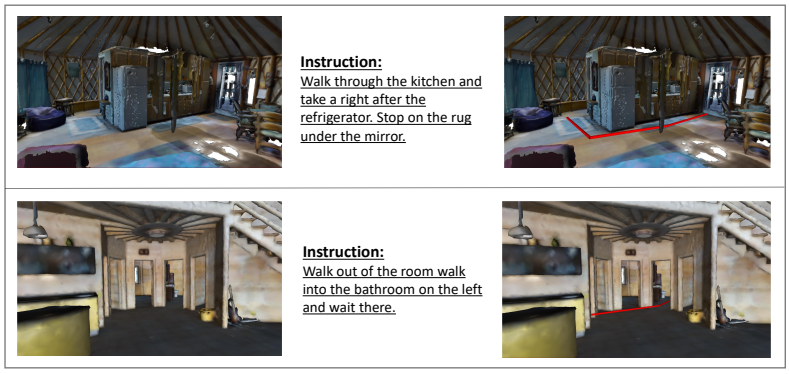


Fig. 9: Examples of 3D vision language navigation. The 3D scene is depicted in the left column, instructions are presented in the middle column, and annotations for the vision language navigation task are shown in the right column. Annotations are stored in textual format, and for visual representation here, we extract the pathway from the text.

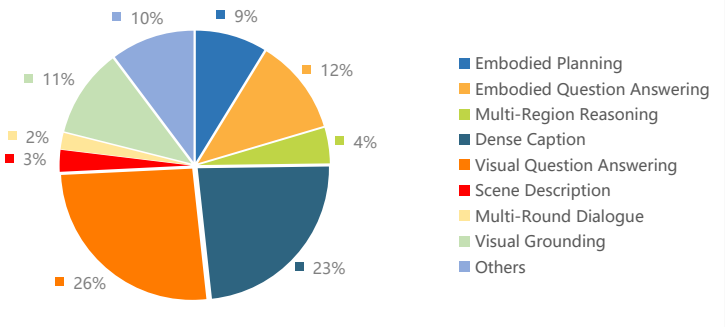
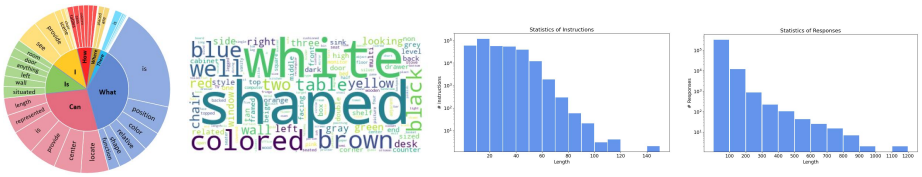


Fig. 10: The evaluation dataset covers a range of fundamental abilities within real-world 3D environments, such as visual perception, scene comprehension, spatial reasoning, and planning.

dimensions such as visual perception, scene understanding, spatial reasoning, and planning. Additionally, we check the quality of each instance, guaranteeing that instructions accurately describe valid tasks and that answers are precise. Our evaluation dataset consists of over 1.5K data samples, distributed as shown in Fig. 10.



(a) Instruction distribution (b) Word cloud of responses (c) # of words in instructions (d) # of words in responses

Fig. 11: The statistics of the M3DBench. (a) The distribution of instructions based on the first word, where the inner circle of the graph represents the frequency of the first word's occurrence, and the outer circle shows the frequency of verbs and nouns appearing in the instructions corresponding to that first word. (b) The word cloud of responses. (c) The distribution of instruction length. (d) The distribution of response length.

1.4 Dataset Statistics and Analysis

To assess the diversity of generated instructions, we analyze the distribution of instructions based on the first word, as shown in Fig. 11 (a). Specifically, we extract the first word of each instruction and collected instructions starting with that word. Then we parse the instructions using the Natural Language Toolkit [2], performing processes like tokenization and part-of-speech tagging to extract nouns and verbs from instructions. The findings indicate that instructions in M3DBench are diverse, including various types such as "What" (query), "Can" (request),

"Is" (confirmation), "I" (first-person), "Where" (location), and so on. Analyzing the word cloud of responses, as depicted in Fig. 11 (b), we observe answers pertaining to shape, color, count, action, object category, spatial relations, and so on. Furthermore, we demonstrate the distribution of lengths for instructions and responses, as illustrated in Fig. 11 (c) and Fig. 11 (d).

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant, and you are seeing an object in a
3D scene. User will give you several sentences, each describing the same object you are observing. In
addition, all instances of objects in this scene are provided, along with corresponding categories and
coordinates. These coordinates are in the form of bounding boxes, represented as [cx, cy, cz, lx, ly, lz] with
floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center
and length of bounding box along x, y, z axis.

Summary and describe the target object in a detail manner, including details like the object placements,
object attributes, object functions, relative position with the surrounding objects, and so on. Here are the
requirements: 1) Describe using the tone of seeing the target object and surroundings. Don't generate
descriptions that cannot be reasoned based on the given information confidently. 2) Descriptions should be
concise, effective, diverse and logical. 3) Do not mention any specific spatial coordinate values and do not
mention the source of information. The description should be more than 100 words and less than 150
words."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n' .join(query)}
```

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant, and you are seeing an object in a
3D scene. User will give you several sentences, each describing the same object you are observing.

Summary the target object in a brief manner, only containing the object attribute. Here are the
requirements: 1) Describe using the tone of seeing the target object. Don't generate descriptions that cannot
be reasoned based on the given information confidently. 2) Descriptions should be concise, effective, and
logical. 3) Do not mention any specific spatial coordinate values and do not mention the source of
information. The description should be less than 5 words."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n' .join(query)}
```

Table 1: System message used to generate detailed (top) and brief (bottom) dense caption data in M3DBench.

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant, and you are seeing an object in a 3D scene. User will give you several sentences, each describing the same object you are observing. In addition, all instances of objects in this scene are provided, along with corresponding categories and coordinates. These coordinates are in the form of bounding boxes, represented as [cx, cy, cz, lx, ly, lz] with floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center and length of bounding box along x, y, z axis.

Generate some questions and give corresponding answer of the target object, including details like the object placements, object attributes, object functions, relative position with the surrounding objects, and so on.  Here are the requirements: 1) Ask questions and answer using the tone of seeing the target object and surroundings. Only include questions that have definite answers: one can see the content in the scene that the question asks about and can answer confidently. Do not ask any question that cannot be answered confidently. Do not ask about uncertain details. 2) Replace the specific target object's name in the question with the placeholder '<target>'. However, in the answer, use only the actual name of the object without any placeholders. 3) Ask diverse questions (e.g., 'What...', 'How...', 'Which...', 'Where...', 'If...', 'Is...', 'Are...', etc.) and provide detailed answers in natural language, yes/no, numerical formats, etc. 4) Ensure questions and answers are concise, logical and effective. 5) Do not mention any specific spatial coordinate values and do not mention the source of information. Keep each question or answer under 50 words."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n'.join(query)}
```

Table 2: System message used to generate instruction-response pairs for visual question answering in M3DBench.

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant, and you are seeing an object in a 3D scene. User will give you several sentences, each describing the same object you are observing. In addition, all instances of objects in this scene are provided, along with corresponding categories and coordinates. These coordinates are in the form of bounding boxes, represented as [cx, cy, cz, lx, ly, lz] with floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center and length of bounding box along x, y, z axis.

You need to generate embodied questions (e.g. Standing in front of the <target> and facing the towels. Can I see myself in the mirror?) and give a corresponding answer. Assuming you are positioned at the target object and facing a nearby object, please begin each question by describing the situation (position, orientation, etc.). Then provide the corresponding answer for the question. Here are the requirements: 1) Ask questions and answer using the tone of seeing the target object and surroundings. Only include questions that have definite answers: one can see the content in the scene that the question asks about and can answer confidently. Do not ask any question that cannot be answered confidently. Do not ask about uncertain details. 2) Replace the specific target object's name in the question with the placeholder '<target>'. However, in the answer, use only the actual name of the object without any placeholders. 3) Ask diverse questions (e.g., 'What...', 'How...', 'Which...', 'Where...', 'If...', 'Is...', 'Are...', etc.) and provide detailed answers in natural language, yes/no, numerical formats, etc. 4) Ensure questions and answers are concise, logical and effective. 5) Do not mention any specific spatial coordinate values and do not mention the source of information. Keep each question or answer under 50 words."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n'.join(query)}
```

Table 3: System message used to generate instruction-response pairs for embodied question answering in M3DBench.

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant, and you are seeing an object in a 3D scene. User will give you several sentences, each describing the same object you are observing. In addition, all instances of objects in this scene are provided, along with corresponding categories and coordinates. These coordinates are in the form of bounding boxes, represented as [cx, cy, cz, lx, ly, lz] with floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center and length of bounding box along x, y, z axis.

Generate some questions and give corresponding answer based on the relationship between these objects. Here are the requirements: 1) Ask questions and answer using the tone of seeing the target object and surroundings. Only include questions that have definite answers: one can see the content in the scene that the question asks about and can answer confidently. Do not ask any question that cannot be answered confidently. Do not ask about uncertain details. 2) Replace the names of related objects in the question with placeholders like '<region 1>', '<region 2>', '<region 3>', etc. However, in the answer, use only the actual names of the objects without any placeholders. 3) Ask diverse questions (e.g., 'What...', 'How...', 'Which...', 'Where...', 'If...', 'Is...', 'Are...', etc.) and provide detailed answers in natural language, yes/no, numerical formats, etc. 4) Ensure questions and answers are concise, logical and effective. 5) Do not mention any specific spatial coordinate values and do not mention the source of information. Keep each question or answer under 50 words."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n' .join(query)}
```

Table 4: System message used to generate instruction-response pairs for multi-region reasoning in M3DBench.

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant that can analyze a 3D scene. User will give you several sentences, each describing the same scene you are observing. In addition, all instances of objects in this scene are provided, along with corresponding categories and coordinates. These coordinates are in the form of bounding boxes, represented as [cx, cy, cz, lx, ly, lz] with floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center and length of bounding box along x, y, z axis.

Summary and describe the scene in a detail manner, including details like the scenario, scene types, room functions, object types, object counts, object locations, object attributes, relative relationships between the objects, and so on. Here are the requirements: 1) You should design a question before describing the scene. The question should be 1 to 2 sentences long. The type of question should be diverse. Either an imperative sentence or a question is permitted. For example, describe the scene in detail.2) Describe using the tone of seeing the whole scene. Don't generate descriptions that cannot be reasoned based on the given information confidently. 3) Descriptions should be concise, effective, diverse and logical. 4) Do not mention any specific spatial coordinate values and do not mention the source of information. The description should be more than 200 words and less than 250 words."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n' .join(query)}
```

Table 5: System message used to generate instruction-response pairs for scene description in M3DBench.

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant that can analyze a 3D scene. User will give you several sentences, each describing the same scene you are observing. In addition, all instances of objects in this 3D are provided, along with corresponding categories and coordinates. These coordinates are in the form of bounding boxes, represented as [cx, cy, cz, lx, ly, lz] with floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center and length of bounding box along x, y, z axis.

Design a conversation between a human and you discussing various aspects related to the scene. Include topics such as the given scenario, room functionality, type of scene, object categories, object counts, their respective locations, attributes, relationships between objects, and so on. Here are the requirements: 1) Both the human and you should discuss the scene in the tone of seeing the whole scene. Use a variety of sentence structures in the conversation. Avoid discussing details that cannot be confidently answered or are uncertain. 2) Initiate the conversation by choosing a specific topic. Ensure the conversation flows naturally and covers a wide range of details while maintaining coherence. 3) Do not mention any specific spatial coordinate values and do not mention the source of information. Each conversation should take at least 5 rounds."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n' .join(query)}
```

Table 6: System message used to generate instruction-response pairs for multi-round dialogue in M3DBench.

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant, and you are seeing an object in a 3D scene. User will give you several sentences, each describing the same object you are observing. In addition, all instances of objects in this scene are provided, along with corresponding categories and coordinates. These coordinates are in the form of bounding boxes, represented as [cx, cy, cz, lx, ly, lz] with floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center and length of bounding box along x, y, z axis.

Generate embodied questions, including planning(e.g. I feel tired/I want to study and where should I go next?), navigation (e.g. how to go from the <target> to the position of bed?). Assuming you are positioned at the target object and facing a nearby object, please begin each question by describing the situation (position, orientation, etc.). Then provide the corresponding answer for the question. Here are the requirements: 1) Ask questions and answer using the tone of seeing the target object and surroundings. Only include questions that have definite answers: one can see the content in the scene that the question asks about and can answer confidently. Do not ask any question that cannot be answered confidently. Do not ask about uncertain details. 2) Replace the specific target object's name in the question with the placeholder '<target>'. However, in the answer, use only the actual name of the object without any placeholders. 3) Ask diverse questions (e.g., 'What...', 'How...', 'Which...', 'Where...', 'If...', 'Is...', 'Are...', etc.) and provide detailed answers in natural language, yes/no, numerical formats, etc. 4) Ensure questions and answers are concise, logical and effective. 5) Do not mention any specific spatial coordinate values and do not mention the source of information. Keep each question or answer under 50 words."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content": '\n' .join(query)}
```

Table 7: System message used to generate instruction-response pairs for embodied planning in M3DBench.

```
messages = [ {"role":"system", "content": f"""You are an AI visual assistant, and you are seeing an object in a 3D scene. User will give you a sentence, describing an object you are observing. In addition, category and coordinates of the target object are given. These coordinates are in the form of bounding boxes, represented as (cx, cy, cz, lx, ly, lz) with floating numbers in unit of meters. These values correspond to the x, y, z coordinates of bounding box center and length of bounding box along x, y, z axis.

Design instruction for point cloud visual grounding task based on the information provided at first and then generate response using the provided bounding box coordinates. Here are the requirements: 1) The instruction should be related to recognizing the target instance in a point cloud. 2) The instruction should include the format of the object, and your response should follow the given format. 3) Replace the specific target object's name in the question with the placeholder '<target>'. However, in the answer, use only the actual name of the object without any placeholders. 4) The response should include the target object's category and coordinate. Don't change the category name or coordinates. Don't add objects that don't exist."""}
]
for sample in fewshot_samples:
    messages.append({"role":"user", "content":sample['context']})
    messages.append({"role":"assistant", "content":sample['response']} )
messages.append({"role":"user", "content":  '\n'  .join(query)}
```

Table 8: System message used to generate instruction-response pairs for visual grounding in M3DBench.

- Describe the <target> concisely.
- Can you provide a brief overview of the <target>?
- Provide a brief description of the given <target>.
- Can you relay a brief and clear account of the <target>?
- Offer a clear and concise depiction of the <target>.
- Sum up the main aspects of the <target> succinctly.
- In brief, how can you depict the <target>?
- Could you share a short summary of the <target>'s features?
- Summarize the visual content of the <target>.
- What's your concise interpretation of the <target>?
- In a short description, what is the <target>?
- Convey a brief description of the essential features of the <target>.
- How would you describe the <target> in brief?

- Describe the following <target> in detail.
- Provide a detailed description of the given <target>.
- Offer a thorough analysis of the <target>.
- Clarify the contents of the displayed <target> with great detail.
- Analyze the <target> in a comprehensive and detailed manner.
- I would appreciate a full and detailed explanation of the <target>.
- I'm interested in a detailed exploration of the <target>; could you provide that?
- Can you dissect the <target>, giving us a comprehensive understanding?
- Share a rich and detailed narrative of the <target>.
- Offer a profound and comprehensive insight into the <target>.

Table 9: Some examples of instructions for brief (top) and detailed (bottom) dense caption in M3DBench.

- Can you determine the category and location of each object found within the point cloud? The output should be a list of tuples in the format (c, x, y, z, l, w, h), where c represents the class label, while x, y, z denote the bounding box center coordinates, and l, w, h indicate its size.
- Identify the object types within the point cloud and deliver tuples (c, x, y, z, l, w, h) based on spatial data. The format of the result should comprise tuples (c, x, y, z, l, w, h), where c denotes the class label, and x, y, z represent the center coordinates of the bounding box, while l, w, h indicate its dimensions.
- Review the point cloud and list all detected objects, including information on their types and locations. Output a list of tuples: (c, x, y, z, l, w, h) where c is the class label, and x, y, z represent the center coordinates, while l, w, h denote the.
- Can you classify and locate objects within the point cloud? Provide (c, x, y, z, l, w, h) tuples for each. Generate a result in the format of tuples (c, x, y, z, l, w, h), where c signifies the class label, and x, y, z denote the center coordinates of the bounding box, while l, w, h represent its size.
- Using the data from the point cloud, identify the type of each object and provide its spatial coordinates. Generate a list of tuples (c, x, y, z, l, w, h), where c is the class label, and x, y, z represent the bounding box center coordinates, while l, w, h denote its size.
- From the point cloud data, extract object types and spatial coordinates as (c, x, y, z, l, w, h) tuples. The result should be structured as tuples (c, x, y, z, l, w, h), where c represents the class label, and x, y, z indicate the bounding box's center coordinates, while l, w, h specify its dimensions.

- Positioned at the <bbox> location within the point cloud, an object within the <class> category can be observed.
- The point cloud includes an object at the <bbox> position, which can be classified under the category of <class>.
- At the <bbox> position in the point cloud, there is an item categorized as <class>.
- The <bbox> position of the point cloud allows for the identification of an object that belongs to the <class> category.
- Within the point cloud, an object classified as <class> is situated at the <bbox> position.
- An object that can be classified as <class> is located at the <bbox> position within the point cloud.
- The <bbox> of the point cloud reveals the presence of an object categorized as <class>.
- At the <bbox> position within the point cloud, there exists an object that falls under the <class> category.
- The point cloud contains an object at the <bbox> position, which can be identified as <class>.

Table 10: Some examples of question (top) and answer (bottom) templates for 3D object detection in M3DBench.

Replace <target> with click prompt:

f"{object_name} close to the pointed spot <click>{x},{y},{z}</click>"
f"{object_name} situated near the pointed point <click>{x},{y},{z}</click>"
f"{object_name} positioned close to the pointed location <click>{x},{y},{z}</click>"
f"{object_name} near the pointed point <click>{x},{y},{z}</click>"
f"{object_name} close to the pointed location <click>{x},{y},{z}</click>"
f"{object_name} situated near the given point <click>{x},{y},{z}</click>"
f"{object_name} situated close to the pointed location <click>{x},{y},{z}</click>"
f"{object_name} positioned near the pointed point <click>{x},{y},{z}</click>"

Replace <target> with box prompt:

f"{object_name} in the region <region>{x},{y},{z},{l},{w},{h}</region>
f"{object_name} situated in the region <region>{x},{y},{z},{l},{w},{h}</region>
f"{object_name} inside the area <region>{x},{y},{z},{l},{w},{h}</region>
f"{object_name} placed in the region <region>{x},{y},{z},{l},{w},{h}</region>
f"{object_name} with center at [{cx}, {cy}, {cz}] and dimensions [{lx}, {ly}, {lz}]",
f"{object_name} positioned at center [{cx}, {cy}, {cz}] with size [{lx}, {ly}, {lz}]",
f"{object_name} centered at [{cx}, {cy}, {cz}] with measurements [{lx}, {ly}, {lz}]",
f"{object_name} having central coordinates [{cx}, {cy}, {cz}] and measurements of [{lx}, {ly}, {lz}]",

Replace <target> with image prompt:

f"<image>{image_id}</image>"

Replace <target> with 3d shape prompt:

f"<shape>{shape_id}</shape>"

Table 11: The formula for interleaved multi-modal instruction generation in M3DBench.

2 Additional Experiments

Quantitative Evaluation on Scene Description. We score the conversational abilities of the baseline models using GPT-4 [10]. As shown in Tab. 12, regarding detailed description capabilities, utilizing pre-trained Transformer [14] as the scene encoder and LLaMA-2-7B [13] as the language decoder yielded the best performance, surpassing the sub-optimal variant by a margin of +1.79 points. Another observation is that all variants based on OPT [17] demonstrated relatively lower performance. In Sec. 5, we provide prompts used for scoring detailed descriptions with GPT-4 [10], along with qualitative results and the GPT-4 [10] scoring criteria.

Quantitative Evaluation on Multi-round Dialogue. For multi-round dialogue, the baseline model employing Transformer [14] as the scene encoder and LLaMA-2-7B [13] as the language decoder demonstrates the optimal performance, surpassing the next best variant by +1.72 points. Similar to the conclusions derived from the results of detailed description, all OPT-based variants [17] exhibit relatively lower performance.

Quantitative Evaluation on 3D Object Localization. For the 3D object localization task (i.e., finding objects in the scene that best match the given instructions), we propose using a unified output format to represent object position. To acquire localization data, we derive 3D bounding boxes from the "[cx, cy, cz, l, w, h]" provided in the generated text. Here, cx, cy, cz correspond to the x, y, z coordinates of the bounding box center, while l, w, h represent the size of the bounding box along the x, y, z axes. For each value defining the

Table 12: Benchmark for scene description. In practice, we feed both responses generated by variant models and labels into GPT-4 and prompt GPT-4 for comparative analysis and scoring.

3D Vision Encoder	LLM Decoder	Score
Pointnet++ [11]	OPT-6.7B [17]	21.27
	LLaMA-2-7B [13]	28.45
Transformer [14]	OPT-6.7B [17]	27.19
	LLaMA-2-7B [13]	**30.24**

Table 13: Benchmark for multi-round dialogue. *Score* is also generated by the GPT-4 [10], based on the evaluation of the model's response.

3D Vision Encoder	LLM Decoder	Score
Pointnet++ [11]	OPT-6.7B [17]	32.46
	LLaMA-2-7B [13]	35.24
Transformer [14]	OPT-6.7B [17]	35.71
	LLaMA-2-7B [13]	**37.43**

Table 14: Benchmark for object localization. We assess the baseline model's ability to identify and localize objects in the 3D scene. Specifically, the baseline model is tasked with outputting the location of the target object a given specific instruction.

3D Vision Encoder	LLM Decoder	Acc@0.25IoU
Pointnet++ [11]	OPT-6.7B [17]	3.09
	LLaMA-2-7B [13]	1.60
Transformer [14]	OPT-6.7B [17]	1.22
	LLaMA-2-7B [13]	**3.57**

3D bounding box, we retain one decimal place. In Tab. 14, we present baseline performances regarding 3D localization. Results indicate that our baseline model exhibits sub-optimal performance on localizing. We leave the improvement of MLMs' abilities in 3D scene perception and localization for future work.

3 Additional Insightful Analyses

Scaling Model. In this study, we investigate the scalability of language model parameters, ranging from 125 million to 6.7 billion, and analyze the impact of model capacity on 3D tasks. Throughout the experiments, we employ a pre-trained Transformer [14] as the scene encoder with its size constant. As illustrated in Fig. 12, the results indicate that OPT-6.7B [17] significantly outperformed alternative models, maintaining superior performance across all examined tasks.

Scaling Data. We analyze how changing the size of the instruction-following dataset impacts an architecture (Fig. 13). This architecture consists of a pre-trained transformer [14] as the encoder and an opt-based [17] model as the decoder. We explore datasets comprising 12.5%, 25%, 50%, and 100% of the total available data. The findings demonstrate that as the dataset size grows, our baseline models consistently enhance their performance across a wide range of tasks.

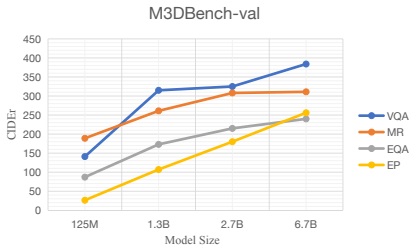


Fig. 12: Scaling Model. As the size of the language models increases from 125 million to 6.7 billion parameters, there is a significant improvement in the performance of the baseline models across various tasks.

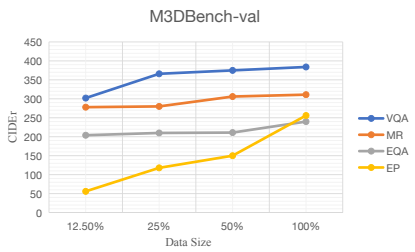


Fig. 13: Scaling Data. When using a fixed model, we find that the baseline consistently improves across diverse tasks as the dataset size increases.

4 Implementation

Scene Encoder. As introduced in **Experiments**, we employ two commonly used types of 3D pre-trained feature extractors as scene encoders: one based on PointNet++ [11] and the other based on Transformer [14]. The PointNet++-based scene encoder comprises four layers for feature extraction and down-sampling, coupled with two layers for feature aggregation and up-sampling [18]. The final layer generates features for sampled points, from which we derive scene-level 256-dimensional features via global max-pooling. In addition, the Transformer-based encoder initially tokenizes input point clouds into 2048 point tokens through a set abstraction layer [11], followed by three cascaded Transformer encoder blocks with masking radii of 0.16, 0.64, and 1.44 [6]. Between the first two Transformer blocks, there is an additional set abstraction layer that downsamples the encoded tokens, with each token represented by 256 dimensions.

Multi-modal Prompt Encoder. We utilize the tokenizer and word embedding from pre-trained LLM [17, 13] to process text instructions. For image inputs, we employed the pre-trained ViT-L/14 [12] as the image encoder, adopting a trainable projector based on the LLaVA [9] to collect image tokens. Regarding 3D shape inputs, we utilized a pre-trained 3D encoder [6] to extract object features, obtaining object-level tokens via another projector. For user's click and pointed region, we directly employ linear layers to project the corresponding prompt features into the LLM embedding space. We leave the design of more optimal models for exploration in future work.

5 GPT-4 Evaluation

We employ the template in Tab. 15 to prompt GPT-4 [10] and obtain corresponding evaluation results. Specifically, we prompt GPT-4 [10] with four inputs: system

message, question, reference answer, and models' responses, which comprise answers from various baseline models. We instruct GPT-4 [10] to assess responses for accuracy, relevance, descriptive details, etc., and assign scores within a range of 0 to 100. Higher scores indicate better quality of the responses. Moreover, we request GPT-4 [10] to provide explanations for the scoring results, assisting in our evaluation of the scoring results' validity. In Fig. 14 and Fig. 15, we present model's (utilizing transformer [14] as the scene encoder and LLaMA [13] as the language decoder) responses, GPT-4 scores, and GPT-4 justifications on detailed description and multi-round dialogue.

[System Message] You are an expert in scene details description. We will provide you a question, a reference answer for this question, and responses from several AI assistants. We would like your feedback on the performance of the AI assistants on their responses to the question. You need to evaluate each assistant's response based on the question and reference answer. Please rate them on accuracy, relevance, naturalness, level of detail of their answers, and correspondence score from 0 to 100. Higher scores indicate better overall quality of the model's response. Please first output the first line indicating the scores for AI assistants, respectively. In the next line, provide a comprehensive explanation of your evaluation, avoiding any potential bias, and ensuring that the order in which the responses were presented does not affect your judgment.

Table 15: The template employed to prompt GPT-4 for assessing the quality of responses across different models.

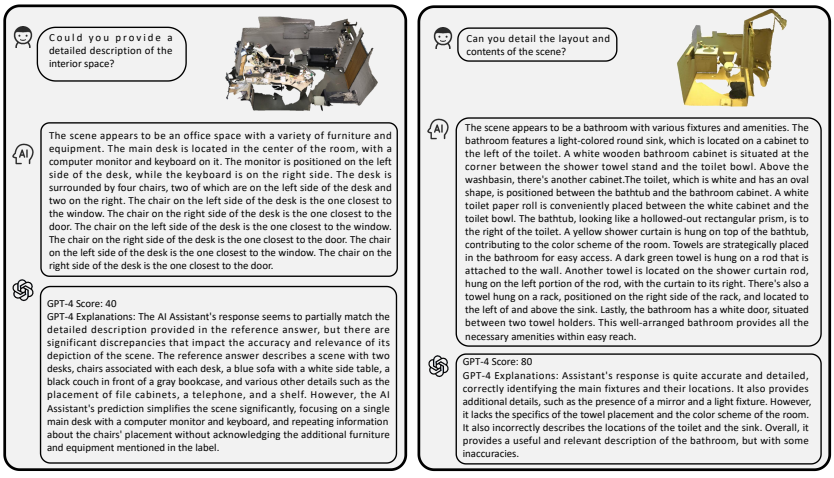


Fig. 14: Qualitative results on scene description. We present visualization results for scene description, along with the scores provided by GPT-4 and the criteria used by GPT-4 for scoring.

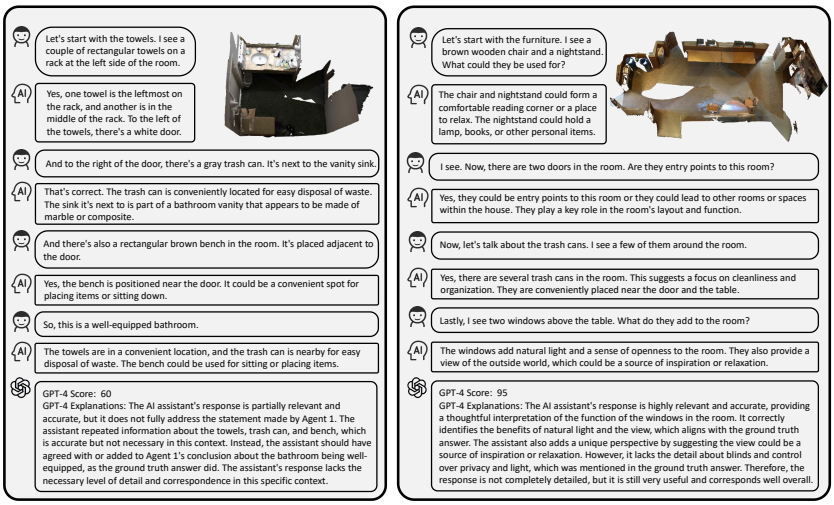


Fig. 15: Qualitative results on multi-round dialogue. We present visualization results for multi-round dialogue, along with the scores provided by GPT-4 and the criteria used by GPT-4 for scoring.

References

1. Achlioptas, P., Abdelreheem, A., Xia, F., Elhoseiny, M., Guibas, L.: Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. pp. 422–440. Springer (2020)
2. Bird, S.: Nltk: the natural language toolkit. In: Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions. pp. 69–72 (2006)
3. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3d: Learning from rgb-d data in indoor environments. arXiv preprint arXiv:1709.06158 (2017)
4. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
5. Chen, D.Z., Chang, A.X., Nießner, M.: Scanrefer: 3d object localization in rgb-d scans using natural language. In: European conference on computer vision. pp. 202–221. Springer (2020)
6. Chen, S., Zhu, H., Chen, X., Lei, Y., Yu, G., Chen, T.: End-to-end 3d dense captioning with vote2cap-detr. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11124–11133 (2023)
7. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017)
8. Krantz, J., Wijmans, E., Majumdar, A., Batra, D., Lee, S.: Beyond the nav-graph: Vision-and-language navigation in continuous environments. In: Computer

Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16. pp. 104–120. Springer (2020)
9. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. arXiv preprint arXiv:2304.08485 (2023)
10. OpenAI, R.: Gpt-4 technical report. arxiv 2303.08774. View in Article **2** (2023)
11. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
12. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
13. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
15. Yin, Z., Wang, J., Cao, J., Shi, Z., Liu, D., Li, M., Sheng, L., Bai, L., Huang, X., Wang, Z., et al.: Lamm: Language-assisted multi-modal instruction-tuning dataset, framework, and benchmark. arXiv preprint arXiv:2306.06687 (2023)
16. Yuan, Z., Yan, X., Li, Z., Li, X., Guo, Y., Cui, S., Li, Z.: Toward explainable and fine-grained 3d grounding through referring textual phrases. arXiv preprint arXiv:2207.01821 (2022)
17. Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X.V., et al.: Opt: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068 (2022)
18. Zhang, Z., Girdhar, R., Joulin, A., Misra, I.: Self-supervised pretraining of 3d features on any point-cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10252–10263 (2021)