6. Appendix

6.1. Additional Experiments and Details

Spatial VQA Human Annotated Benchmark We manually labelled 546 qualitative and quantitative question pairs for WebLi and robotic manipulation VQA. In the human annotation pipeline, we use the spatial VQA data generation pipeline described in Section 3 to provide a sample question for each image, the human annotator would look at the image and the sample question, decide if he or she would like to use the question or type a more proper question, or to skip the annotation for the image. Then based on the sample question or the human input question, the human annotator would type the answer he or she thinks as proper in the form of natural language. Fig. 7, Fig. 8, and Fig. 9 shows examples of the human annotated spatial VQA pairs.

Chain-of-thoughts Here we provide more details to our implementation of chain-of-thought spatial reasoning. As we mentioned in main paper, we prompt a LLM to perform chain-of-thought reasoning with ability to query our SpatialVLM for visual information. Since the LLM isn't aware of visual information itself, we prompt it to make decision as if it's playing a game, by asking its friend who can see an image that it cannot see itself. We provide the full prompt below:

Listing 1. SpatialVLM CoT Prompts

```
You are participating in a visual question answering game
      with your friend. In this game, you are presented
      with a question which requires visual information
      from an image to answer. You can see the question
      but not the image, while your friend can see
     the image but not the original question. Luckily, you
      are allowed to decompose the question and ask your
     friend about the image. Your friend gives you answers
      which can be used to answer the original question.

Here is a sample conversation:
[Question] How can I clean up the table? Give
     detailed instruction about how should I move my hand.
[You] What objects are there in the image?
[Friend] There is
      an empty coke can, a trash bin and a coffee machine.
[You] Is the trash
      bin to the left or to the right of the coke can?
[Friend] It's to the left.
[You] Is the trash bin or the coke can further from you?
[Friend] They are similar in depth.
[You] How much to
      the left is the trash bin compared to the coke can?
[Friend] Around 20 centimeters.
[Answer]
      One should grab the coke can, move it 20 centimeters
      left and release it so it falls in the trash bin.

Here is another example:
[Question] Tell me if the distance
      between the blue bottle and the yellow book is
      longer than that between the plant and the coke can?
[You] What is the
     distance between the blue bottle and the yellow book?
[Tool] 0.3m
[You] What
      is the distance between the plant and the coke can?
[Friend] 0.7m
[Robot] Since the distance between the blue bottle
      and the yellow book is 0.3m and distance between
      the plant while the coke can is 0.7m, the distance
      between the blue bottle and the yellow book is not
      longer than that between the plant and the coke can.
[Answer] No.

Here is another example:
[Question] Which object
      can be reached by kids more easily, the white and
      yellow rabbit toy can or the dark green can of beer?
[You] What is
     the elevation of the white and yellow rabbit toy can?
[Friend] 0.9 m.
[You] What is the elevation of the dark green can of beer?
[Friend] 0.2 m.
[Answer] Since
     the kids are generally shorter, it is easier for them
      to reach something that are lower in altitude, so
     it would be easier for them to reach the can of beer.

Now, given a new question, try to answer the questions
     by asking your friend for related visual information.
[Question]
```

By doing so, we find LLM and SpatialVLM can effectively work together to derive the correct .

6.2. Implementation Details

Semantic Filtering In the data filtering phase, we have 2 important objectives: First, we shall filter out images that humans can hardly ask any spatial questions, such as photo of a single object before a white background. Second, since our process requires lifting a 2D image to 3D point cloud, we desire the field of view to be close to a value our monocular depth estimation model is optimized for.

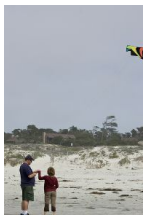
To achieve the first objective, we use a pretrained CLIP model to label the candidate images, and filter out those that represent a product or an artwork. Positive CLIP labels include "an iphone photo of an indoor scene", and "an iphone photo of an outdoor scene", while negative labels are "a close up shot of a single object", "a product displayed in front of a white background", "an artwork", "a painting", "a screenshot of graphics user interface", "a piece of text", "a sketch".

We choose "an iphone photo" as a prefix for positive cases to satisfy the second objective. We observe that this prefix effectively filters out data that has a wider field of view, as well as certain images with uncommon perspective ratio.

Such design choices in data filtering ensure the images left are within the effective distribution of our expert models and qa generation.

2D Contexts Extraction As we mentioned in method section, we use a variety of off-the-shelf models to extract relavent information to synthesize our question answer data. Here we provide additional details to context extraction. After data filtering, we run a region proposal network (RPN) followed by a non-max suppression (NMS) [30]. For each object bounding box, we run a class agnostic segmentation model [41] to segment out the object. For each bounding box, we use FlexCap [4] to sample an object-centric caption with random length between $1-6$

Q: Is the fireplace screen with red doors smaller than the dog standing on the floor in width?
A: no



Q: Compared to the little boy in a red shirt, which side is the man wearing a blue shirt on?
A: left

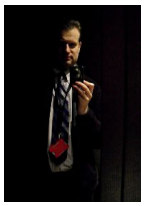
Q: Are the windows positioned to the left of the black television?
A: yes

Q: Is the palm tree in distance taller than the parked white car?
A: yes

Figure 7. **Example question-answer pairs of the Spatial VQA qualitative benchmark**

Q: Could you provide the distance between the sign and the motorcyclist?
A: about 0.5 meter

Q: Determine the distance of the fence from the giraffes in a zoo relative to the camera.
A: about 5 meters



Q: How far is the striped tie towards the left from the black cell phone?
A: about 0.2 meter

Q: What is the distance between the sand and the people that are standing on the beach?
A: 0, as the people are directly standing on the sand

Figure 8. **Example question-answer pairs of the Spatial VQA quantitative benchmark**

Q: How far is the yellow robot gripper finger from the white bottle that is laying on the left side of the table?
A: 10 cm.

Q: What is the elevation of the bag of chips on table with respect to the table top surface?
A: The height of the bag of chips on table from the ground is 6 cm.

Q: How much distance is the red apple from the can on the table?
A: It's approximately 20 centimeters.

Q: Could you provide the distance between the yellow finger and the black snacks bag?
A: 5 inches

Figure 9. **Example question-answer pairs of the robotic manipulation VQA quantitative benchmark**

words. In particular, we deliberately choose to avoid traditional object detectors, as they are fixed to very coarse categories such as "cake", while our approach can annotate objects with fine-grained descriptions like "cake shaped like a house" and "cup cake in plastic container" for the image in Figure 2.

2D Context to 3D Context Lifting We then run the state-of-the-art metric depth detector, ZoeDepth [6] on the image. ZoeDepth outputs metric depth (in real-world "meters"). Combined with an fov estimation, we are able to to lift 2D images into 3D point clouds as illustrated with real-world scale as illustrated in Figure 2.

In this point cloud processing step, outliers, or points that significantly deviate from the main group, are removed to enhance data accuracy. A clustering algorithm DBSCAN [22] groups the points based on proximity, focusing on densely populated regions and eliminating sparse, less significant points. This results in a cleaner, more structured point cloud, ideal for subsequent shape and geometry analysis where dimensions of the shapes are measured. Since we already obtained the semantic segmentation for the point cloud, we may use this information to process outliers at adaptive scales. For smaller objects, we use a smaller threshold, proportional to the along each axis. We observe that such choice effectively removes point cloud outliers while also keeping important points for smaller objects. We provide algorithm details below in Algorithm 2.

Listing 2. Outlier removal with DBScan

```
Input:
```

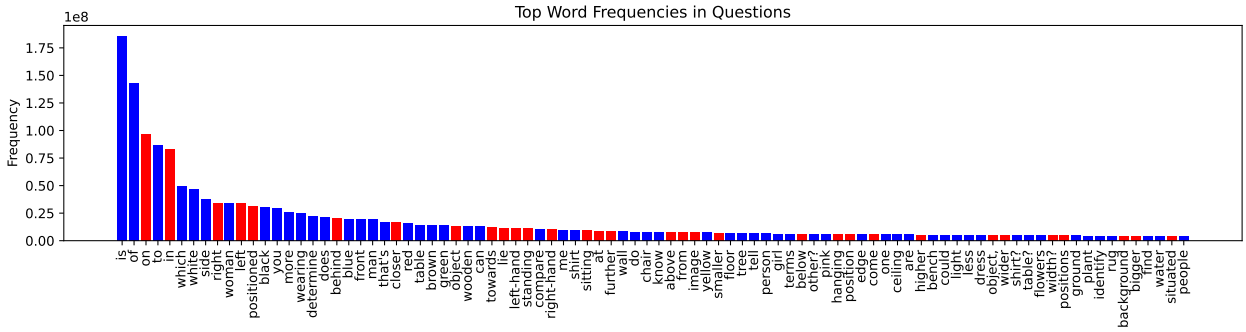


Figure 10. **Top word frequency**. Top words appeared in the training dataset, the red color indicate the word is involved in discussing a spatial concept. It shows that our training data is rich and diverse in spatial questions.

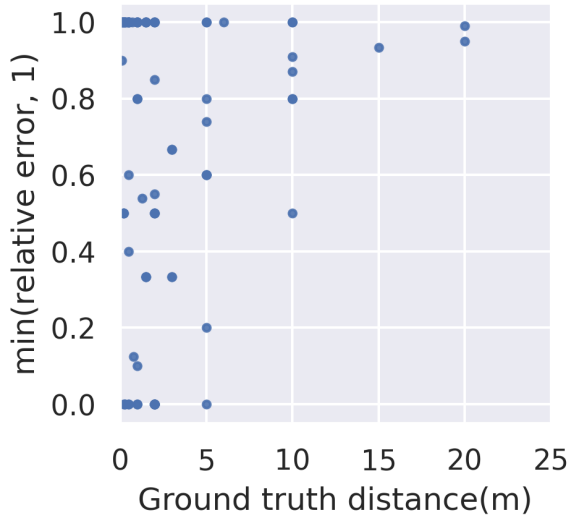


Figure 11. **Error vs Scene Depth ablation.** The errors that SpatialVLM make eventually attributes to the noise in the data, we plot the distance estimation relative error (capped at 1.0) w.r.t. ground truth distance, and found that there are generally larger errors for bigger distance. We hypothesize this might be due to dataset bias of ZoeDepth [6].

```
points_obj: Pointcloud of object of interest
pcd: Full Pointcloud

scale = norm(points_obj.std(axis=0)) * 3.0 + 1e-6
pcd = pcd.remove_stat_outlier(neighbors=50, std=1.2)
pcd = pcd.down_sample(voxel_size=max(0.01, scale / 20))
labels = array(pcd.cluster_dbscan(
   eps=scale / 3.6, min_points=len(pcd) // 10))
```

Coordinate Canonicalization Now we have a 3D point cloud under metric scale. However, the point cloud is still in camera frame, which limits the information we can extract. For example, an object closer to the upper side of the image isn't necessarily further from the ground, because the camera might be pointing at the ground instead of the front. In order to solve this problem, we canonicalize the coordinate system of the pointcloud by detecting horizontal surfaces. We use a light weight segmentation model [10] to segment out pixels correspond to categories like "floor", "table top", before using RANSAC to fit the biggest plane among these 3D points.

When we detect a surface defined by enough points, we canonicalize the coordinate of the point cloud by creating a new origin by projecting camera origin to the detected plane. We use the normal axis of the detected plane as z-axis and project the original z-axis of camera on the the plane as the new x-axis. By doing so, when we can detect horizontal surfaces like ground, we effectively transform the point cloud into world coordinate instead of camera coordinate. A more detailed algorithm can be found in our algorithm box 3. On the other hand, when not enough points corresponding to horizontal surfaces are detected, we flag canonicalization as failed and avoid synthesizing questions that depends on canonicalization, such as questions about elevation.

Listing 3. Canonicalization Algorithm

```
Input:
depth: predicted depth for each point
ground_mask: detected ground or not for each point

points_cam = unproject_to_pointcloud(depth, fov)
points = points_cam
ground_mask = ground_mask.flatten()
canonicalized = False

if ground_mask.mean() > canonicalize_threshold:
  canonicalized = True
  ground_pcd = subset(points_cam, ground_mask)
  plane, _ = ground_pcd.segment_plane(
     distance_threshold
          =0.05, ransac_n=3, num_iterations=1000)
  if array([0, -1, 0]) @ plane[:3] < 0:
    plane = -plane
  a, b, c, d = plane
  normal = array([a, b, c])
  ez = array([0, 0, 1])
  new_y = ez - normal @ ez * normal
  new_y = new_y / norm(new_y)
  rot = array([cross_prod(new_y, normal), new_y, normal])
  rot = array([[0, -1, 0], [1, 0, 0], [0, 0, 1]]) @ rot
  trans = array([0, 0, d])
  points_world = points_cam @ rot.T + trans[None]
  points = points_world

return points, canonicalized
```

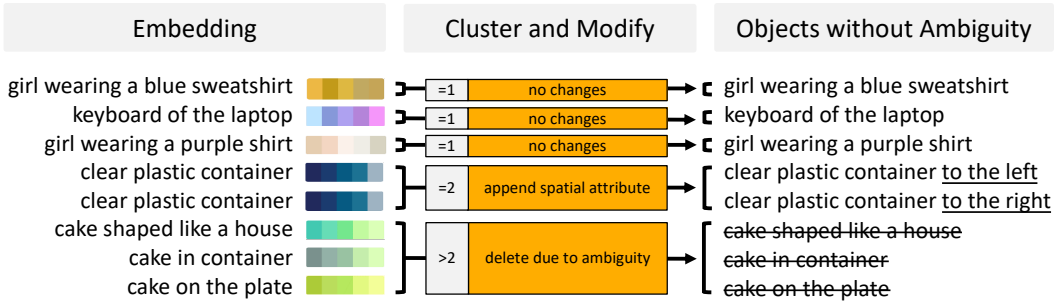


Figure 12. This is a figure illustrating ambiguity removal.

Ambiguity Removal As shown in Figure 12, we first embed all captions with CLIP encoder [52]. This allows us to calculate a cosine distance between each caption pair. This forms a similarity matrix between all objects. If we threshold the similarity score, we can identify whether an object caption is too close to others. In many cases, we have groups of exactly two similar captions, so we can easily augment each caption by appending an differentiating clause such as "that's more to the top of the image". Other cases involve more than two similar captions, which we choose to remove all together to avoid ambiguity. We also remove common background objects based on CLIP similarity to categories like "sun" or "sky".

Human Alignment Humans rarely say a distance measure with many decimal places like 0.95 meters. Rather, they round such distance into some thing they prefer, such as 1 meter or half a meter. We would like our model to align with such human preferences as well. In fact, since depth estimation and fov estimation contain irreducible errors, the model should be allowed to phrase its answers with uncertainty by rounding just like humans, unless when prompted to be accurate. To this end, we post process any quantitative distance unit to align with human preferences. As illustrated in Figure 13, we coded a decision tree to make such alignment. For example, when the estimated distance is 0.86 meters, with 75% probability we just round it to 1 meters, while we answer 3 feet, 90 cm with some lower probabilities. For a distance like 23 meters, we round it to 20 meters with high probability as well. We also sample imperial units instead of metric units by a 20% chance, with similar human-like rounding rules.

While this may align the model better with humans, at sampling time, one may want to get more accurate distance estimation. To do so, one simply need to sample multiple distance estimations and take the average. We used this in our robotic experiments to get more fine-grained values. Another way is to prompt VLM itself to keep a certain amount of digits. This can be added to data synthesis but we leave this to future work.

We found that humans also prefer shorter captions for objects. Therefore, We encourage shorter answers by putting more sampling weight on shorter templates and object captions via a softmax function on negative length. We also coded a additional layer of reweighting to encourage the selection of semantically distinct objects base on CLIP score, as humans tend to ask questions about distinct object types.

VLM Training Our end-to-end training on multi-modal data is from scratch, rather than fine tuning a base VLM that's already trained on multi-modal data. However, just like PaLM-E, the single modal language model and vision encoder are initialized from pre-trained weights respectively. We train our multi-modal large language model with a batch size of 512 and an ADAM optimizer with learning rate of 2e-4. We trained the PaLM 2-E-S model using a mixture of the original VQA datasets in PaLM-E and our generated spatial VQA dataset, with a sampling ratio of *174:2.5*. We initially train the model with a frozen vision encoder for 110k steps, which doesn't use all the data we exhausted. Therefore the data we generated is more than enough. We then, like described in the experiment section, finetune with the vision encoder either frozen or unfrozen for 70k steps till convergence.

6.3. Question and Answer Template

As we mentioned in our method section, we synthesis question and answering pairs via templates. Given a description of a pair of objects, such as "the yellow banana" and "the cake in the shape of a house", our data synthesis pipeline can effectively extract an answer based on question types.

Here we provide a distribution of the question and answer types in Figure 14 and Figure 15, followed by a description about each category.

1. **left predicate** A question asking whether object A is to the left of object B from the viewer's perspective. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
2. **right predicate** A question asking whether object A is to the right of object B from the viewer's perspective. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
3. **above predicate** A question asking whether object A is above object B. Requires coordinate canonicalization. The solution is a binary predicate true or false expressed in

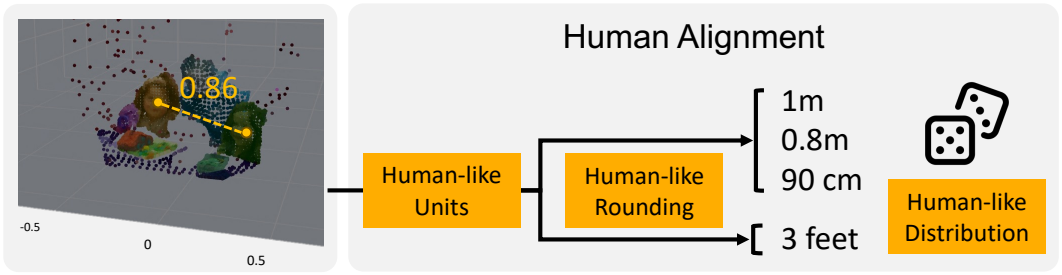


Figure 13. In human alignment, we define a set of rules that rounds with a probability that mimics the decision rules of humans.

natural language, or a phrase expressing uncertainty.

4. **below predicate** A question asking whether object A is below object B. Requires coordinate canonicalization. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
5. **behind predicate** A question asking whether object A behind object B from the viewer's perspective. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
6. **front predicate** A question asking whether object A is in front of object B. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
7. **tall predicate** A question asking whether object A is taller than object B. Requires coordinate canonicalization. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
8. **short predicate** A question asking whether object A is shorter than object B. Requires coordinate canonicalization. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
9. **wide predicate** A question asking whether object A is wider than object B. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
10. **thin predicate** A question asking whether object A is thiner than object B. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
11. **big predicate** A question asking whether object A is bigger than object B. Requires coordinate canonicalization. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
12. **small predicate** A question asking whether object A is smaller than object B. Requires coordinate canonicalization. The solution is a binary predicate true or false expressed in natural language, or a phrase expressing uncertainty.
13. **left choice** A question asking which of object A and object B is more to the left from the viewer's perspective. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
14. **right choice** A question asking which of object A and object B is more to the right from the viewer's perspective. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
15. **above choice** A question asking which of object A and object B is more above. Requires coordinate canonicalization. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
16. **below choice** A question asking which of object A and object B is more below. Requires coordinate canonicalization. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
17. **behind choice** A question asking which of object A and object B is more behind. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
18. **front choice** A question asking which of object A and object B is more to the front from the viewer's perspective. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
19. **tall choice** A question asking which of object A and object B is taller. Requires canonicalization. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
20. **short choice** A question asking which of object A and object B is shorter. Requires canonicalization. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
21. **wide choice** A question asking which of object A and object B is wider. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
22. **thin choice** A question asking which of object A and object B is thinner. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
23. **big choice** A question asking which of object A and object B is bigger. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
24. **small choice** A question asking which of object A and object B is smaller. The solution is an object name expressed in natural language, or a phrase expressing uncertainty.
25. **left-right classify** A question asking about the left-right

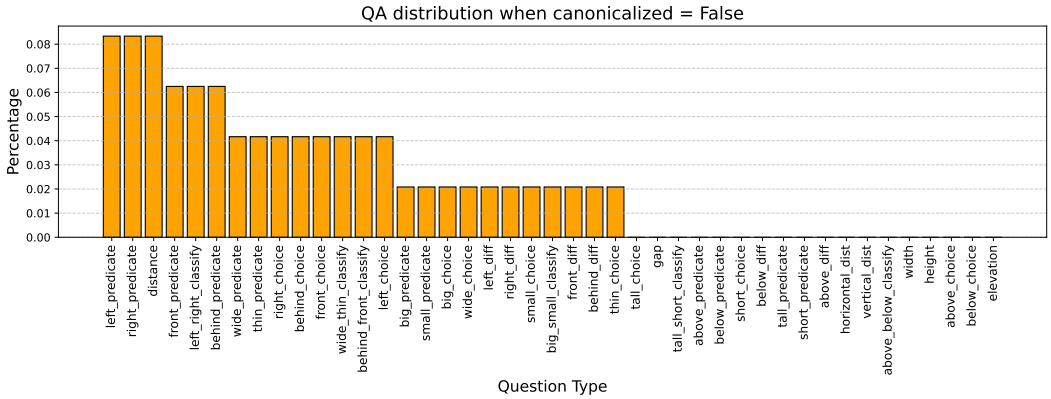


Figure 14. Distribution of generated question-answer categories when canonicalization fails.

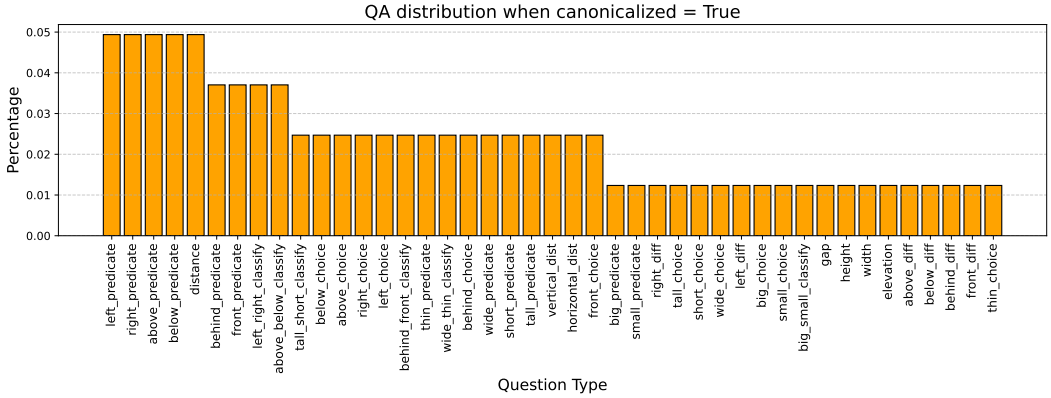


Figure 15. Distribution of generated question-answer categories when canonicalization is successful.

comparative relationship between two objects. The solution is left-right expressed in natural language, or a phrase expressing uncertainty.

26. **above-below classify** A question asking about the above-below comparative relationship between two objects. Requires canonicalization. The solution is above-below expressed in natural language, or a phrase expressing uncertainty.
27. **behind-front classify** A question asking about the behind-front comparative relationship between two objects. The solution is behind-front expressed in natural language, or a phrase expressing uncertainty.
28. **tall-short classify** A question asking about the tall-short comparative relationship between two objects. Requires canonicalization. The solution is tall-short expressed in natural language, or a phrase expressing uncertainty.
29. **wide-thin classify** A question asking about the wide-thin comparative relationship between two objects. The solution is wide-thin expressed in natural language, or a phrase expressing uncertainty.
30. **big-small classify** A question asking about the big-small comparative relationship between two objects. Requires canonicalization. The solution is big-small expressed in natural language, or a phrase expressing uncertainty.
31. **distance estimation** A question asking about the distance between the center of two objects. The solution is a distance expressed in natural language, with a human-like distribution for rounding.
32. **gap estimation** A question asking about the gap between two objects. The solution is a distance expressed in natural language, with a human-like distribution for rounding.
33. **height estimation** A question asking about the height of an object. The solution is a distance expressed in natural language, with a human-like distribution for rounding. Requires canonicalization.
34. **width estimation** A question asking about the width of an object. The solution is a distance expressed in natural language, with a human-like distribution for rounding.
35. **elevation estimation** A question asking about the elevation of an object. The solution is a distance expressed in natural language, with a human-like distribution for rounding. Requires canonicalization.
36. **vertical distance estimation** A question asking about the vertical distance between the center of two objects.

The solution is a distance expressed in natural language, with a human-like distribution for rounding. Requires canonicalization.

37. **horizontal distance estimation** A question asking about the horizontal distance between the center of two objects. The solution is a distance expressed in natural language, with a human-like distribution for rounding. Requires canonicalization.
38. **above difference estimation** A question asking about the the distance between the bottom of more elevated object and the bottom of the less elevated object. The solution is a distance expressed in natural language, with a human-like distribution for rounding. Requires canonicalization.
39. **below difference estimation** A question asking about the distance between the bottom of less elevated object and the bottom of the more elevated object. The solution is a distance expressed in natural language, with a human-like distribution for rounding. Requires canonicalization.
40. **behind difference estimation** A question asking about how much an object is in behind another a distance alone the camera ray. The solution is a distance expressed in natural language, with a human-like distribution for rounding.
41. **front difference estimation** A question asking about how much an object is in in front of another a distance alone the camera ray. The solution is a distance expressed in natural language, with a human-like distribution for rounding.
42. **left difference estimation** A question asking about how much an object is to the left of another, from the viewer's perspective. The solution is a distance expressed in natural language, with a human-like distribution for rounding.
43. **right difference estimation** A question asking about how much an object is to the right of another, from the viewer's perspective. The solution is a distance expressed in natural language, with a human-like distribution for rounding.

We provide a small set of question and answer pairs for generating QA data. For the full list please refer to our website.

Listing 4. SpatialVLM Question and Answer Template

```
OBJ_A = "[A]"
OBJ_B = "[B]"
DIST = "[X]"

distance_questions = [
    "What is the distance between [A] and [B]?",
    "How far apart are [A] and [B]?",
    "How distant is [A] from [B]?",
    "How far is [A] from [B]?",
    "How close is [A] from [B]?",
    "Could you measure the distance between [A] and [B]?",
    "Can you tell me the distance of [A] from [B]?",
    "How far away is [A] from [B]?",
    "Can you provide
          the distance measurement between [A] and [B]?",
    "Can you give me an
          estimation of the distance between [A] and [B]?",
    "Could you provide the distance between [A] and [B]?",
    "How much distance is there between [A] and [B]?",
    "Tell me the distance between [A] and [B].",
    "Give me the distance from [A] to [B].",
    "Measure the distance from [A] to [B].",
    "Measure the distance between [A] and [B].",
]


distance_answers = [
    "[X]",
    "[A] and [B] are [X] apart.",
    "[A] is [X] away from [B].",
    "A distance of [X] exists between [A] and [B].",
    "[A] is [X] from [B].",
    "[A] and [B] are [X] apart from each other.",
    "They are [X] apart.",
    "The distance of [A] from [B] is [X].",
]


vertical_distance_questions = [
    "What is the vertical distance between [A] and [B]?",
    "How far apart are [A] and [B] vertically?",
    "How distant is [A] from [B] vertically?",
    "How far is [A] from [B] vertically?",
    "Could you measure
          the vertical distance between [A] and [B]?",
    "Can you tell
          me the vertical distance between [A] and [B]?",
    "How far away is [A] from [B] vertically?",
    (
        "Can you provide the measurement
              of the vertical distance between [A]"
        " and [B]?"
    ),
    "Estimate the vertical distance between [A] and [B].",
    "Could you provide
          the vertical distance between [A] and [B]?",
    "How much distance
          is there between [A] and [B] vertically?",
    "Tell me the distance between [A] and [B] vertically.",
    "Give me the vertical distance from [A] to [B].",
    "Measure the vertical distance from [A] to [B].",
    "Measure the distance between [A] and [B] vertically.",
]


vertical_distance_answers = [
    "[X]",
    "[A] and [B] are [X] apart vertically.",
    "[A] is [X] away from [B] vertically.",
    "A vertical
          distance of [X] exists between [A] and [B].",
    "[A] is [X] from [B] vertically.",
    "[A] and
           [B] are [X] apart vertically from each other.",
    "Vertically, They are [X] apart.",
    "The vertical distance of [A] from [B] is [X].",
    "They are [X] apart.",
    "It's approximately [X]."
]


horizontal_distance_questions = [
    "What is the horizontal distance between [A] and [B]?",
    "How far apart are [A] and [B] horizontally?",
    "How distant is [A] from [B] horizontally?",
    "How far is [A] from [B] horizontally?",
    "Could you measure
          the horizontal distance between [A] and [B]?",
    "Can you
         tell me the horizontal distance of [A] from [B]?",
    "How far away is [A] from [B] horizontally?",
    (
        "Can you provide the measurement
              of the horizontal distance between [A]"
        " and [B]?"
    ),
    (
        "Can you give me an estimation
              of the horizontal distance between [A]"
        " and [B]?"
    ),
```

```
    "Could you provide
          the horizontal distance between [A] and [B]?",
    "How much distance
          is there between [A] and [B] horizontally?",
    "Tell me
          the distance between [A] and [B] horizontally.",
    "Give me the horizontal distance from [A] to [B].",
    "Vertial gap between [A] and [B].",
    "Measure the horizontal distance from [A] to [B].",
    "Measure
          the distance between [A] and [B] horizontally.",
]


horizontal_distance_answers = [
    "[X]",
    "[A] and [B] are [X] apart horizontally.",
    "[A] is [X] away from [B] horizontally.",
    "A horizontal
          distance of [X] exists between [A] and [B].",
    "[A] is [X] from [B] horizontally.",
    "[A] and
           [B] are [X] apart horizontally from each other.",
    "Horizontally, They are [X] apart.",
    "The horizontal distance of [A] from [B] is [X].",
    "They are [X] apart.",
    "It's approximately [X]."
]

width_questions = [
    "Measure the width of [A].",
    "Determine the horizontal dimensions of [A].",
    "Find out how wide [A] is.",
    "What is the width of [A]?",
    "How wide is [A]?",
    "What are the dimensions of [A] in terms of width?",
    "Could you tell me the horizontal size of [A]?",
    "What is the approximate width of [A]?",
    "How wide is [A]?",
    "How much space does [A] occupy horizontally?",
    "How big is [A]?",
    "How big is [A] in terms of width?",
    "What's the radius of [A]?"
]

width_answers = [
    "[X]",
    "The width of [A] is [X].",
    "[A] is [X] wide.",
    "[A] is [X] in width.",
    "It's [X]."
]

behind_predicate_questions = [
    "Is [A] behind [B]?",
    "Is the
          position of [A] more distant than that of [B]?",
    "Does [A] lie behind [B]?",
    "Is [A] positioned behind [B]?",
    "Is [A] further to camera compared to [B]?",
    "Does [A] come behind [B]?",
    "Is [A] positioned at the back of [B]?",
    "Is [A] further to the viewer compared to [B]?",
]


behind_true = [
    "Yes.",
    "Yes, it is.",
    "Yes, it's behind [B].",
    "That's True.",
    "Yes, [A] is further from the viewer.",
    "Yes, [A] is behind [B]."
]


behind_false = [
    "No.",
    "No, it is not.",
    "No, it's in front of [B].",
    "That's False.",
    "No, [A] is closer to the viewer.",
    "No, [B] is in front of [A]."
]


front_predicate_questions = [
    "Is [A] in front of [B]?",
    "Is the
          position of [A] less distant than that of [B]?",
    "Does [A] lie in front of [B]?",
    "Is [A] positioned in front of [B]?",
    "Is [A] closer to camera compared to [B]?",
    "Does [A] come in front of [B]?",
    "Is [A] positioned before [B]?",
    "Is [A] closer to the viewer compared to [B]?",
]


front_true = [
    "Yes.",
    "Yes, it is.",
    "Yes, it's in front of [B].",
    "That's True.",
    "Yes, [A] is closer to the viewer.",
    "Yes, [A] is in front of [B]."
]


front_false = [
    "No.",
    "No, it is not.",
    "No, it's behind [B].",
    "That's False.",
    "No, [A] is further to the viewer.",
    "No, [B] is behind [A]."
]
```

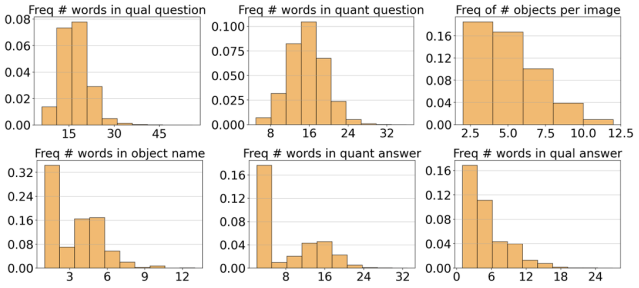


Figure 16. Histogram of # words in object caption and QA.

Number of words The number of words in questions or in object names provide useful insights about the number of tokens and how specific descriptions are. In Figure 16, which illustrates the distribution of the number of words in object captions, questions, and answers. We also plot the number of objects detected in the image.

6.4. Scaling study

We planned to perform a scaling study of SpatialVLM performance against the amount of generated data. However, since SpatialVLM trains a multi-billion parameter multi-modal large language model from scratch, such scaling study is prohibitively expensive during submission to this conference. However, we plan to continue this study in future version of the paper as well as on our website.

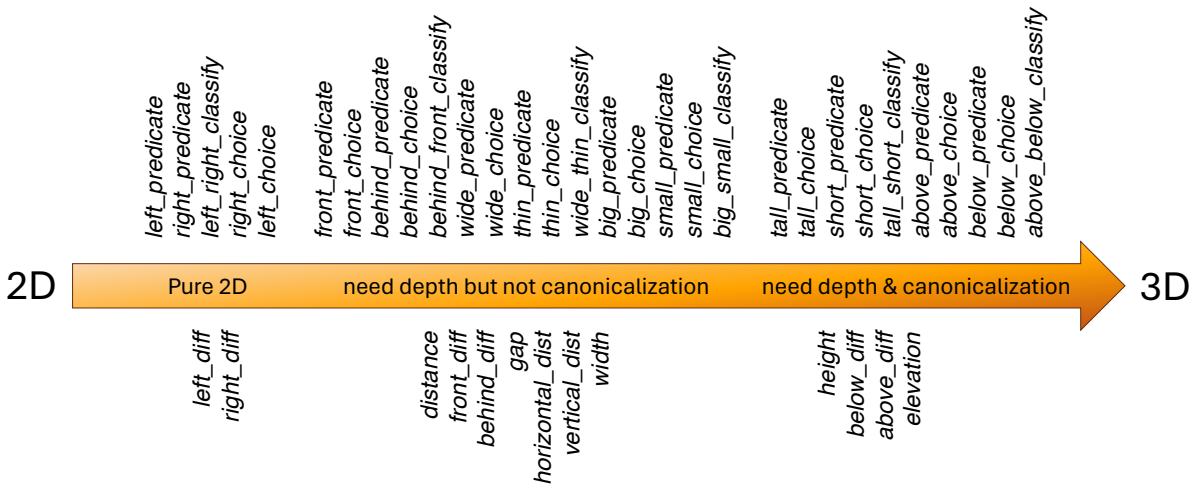


Figure 17. Level of 3D required in different types of question

6.5. Level of 3D required by question type

One interesting question to ask is how important is 3D data vs just 2D data synthesis. In Figure 17, we illustrate the level of 3d awareness required to answer each type of question. In general, we categorize questions into 3 levels. The 2D only level contains only one type, "left-right" . The second level consists of 3D tasks that do not require the model to discern the vertical axis (e.g. "near-far"), while the third level requires understanding of the vertical dimension (e.g. "above-below"). Questions in the third level are all highlighted in their descriptions as "requires canonicalization" in Appendix 6.3 while others are level one and two.

With additional experiments, we found that SpatialVLM achieves an 9% higher success rate in tasks requiring 3D information compared to the best baseline, LLaVA 1.5 (69%), while the gap for more 2D questions is much smaller (2%). This indicates that it is indeed the 3D capability that gives SpatialVLM its edge.

6.6. Filtering percentage

We found that 93.21% of the original data is filtered by CLIP filtering. This demonstrates that noisy internet data must undergo aggressive filtering through an automatic pipeline. We observe that many internet images are product photos or screenshots, rather than scene level data that contains multiple objects.

We found 52.4% of the objects are removed during the ambiguity removal stage. Even when a scene contains multiple objects, many of them are of the same kind, and it's hard for human to refer to a specific instance through image.