Lifting Unlabeled Internet-level Data for 3D Scene Understanding

Supplementary Material

A. Data Curation

As described in Sec. 3, we provide detailed information on how sparse reconstruction data are generated from Internet videos. The raw Internet data are collected from housing-tour videos on YouTube[1] and Bilibili[2], which contain a total of 8,217 videos, from which we obtain 6,687 reconstructed scene instances. The overall data processing pipeline consists of two main stages: preprocessing and reconstruction, as shown in Fig. S.2.

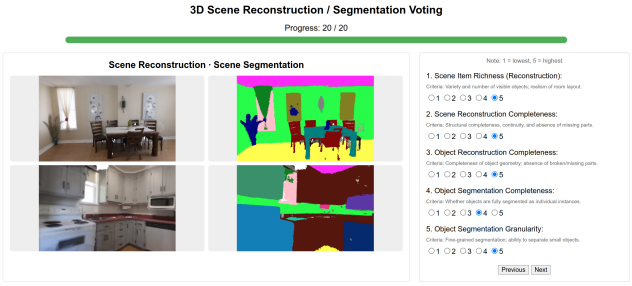
Preprocessing stage Internet videos often contain many shots rather than a continuous shot, which can significantly degrade the reconstruction quality if treated as a single sequence. To address this, we use TransNetV2 [99] to detect the shot boundary and split each video into multiple shots, each treated as an individual scene. Since each clip still includes a large number of redundant or noisy frames, we use parallax-based keyframe selection to retain representative frames, and employ detection models to filter out outdoor frames and frames that contain humans. To ensure both reconstruction efficiency and quality, long sequences are further subdivided based on the number of keyframes, with a maximum clip length of 300 frames and an overlap of 50 frames between adjacent clips.

Reconstruction stage To efficiently establish image correspondences, we use a loop and sequence pairing strategy. In the loop pairing strategy, we extract image features and compute feature distances to other images within a 100-frame range. The top 50 image pairs with feature distances greater than 0.4 are retained as valid loop pairs. In the sequence pairing strategy, the preceding and following 20 frames are used as sequential pairs. We then extract feature points [31] for each image pair and perform feature matching across the selected pairs to generate point correspondences. Finally, we use COLMAP to estimate the camera parameters and complete the sparse reconstruction.

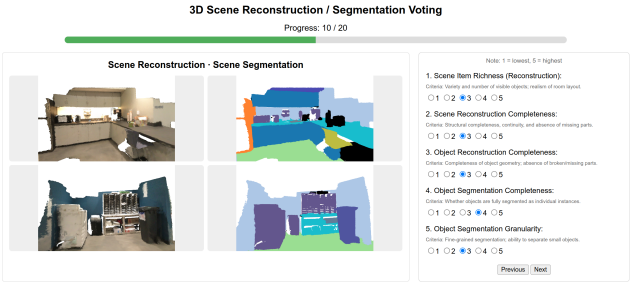
B. Data Quality Check

To assess the quality of data produced by our automated data engine, we perform a human evaluation on the reconstruction and instance segmentation. More specifically, we sample 10 scenes from SceneVerse++ and ScanNet, respectively, visualize their reconstruction and segmentation results side-by-side, and ask human subjects to rate each scene on a scale of 1 to 5, along the following 5 axes:

[1] http://youtube.com/
[2] http://bilibili.com/



(a) An example from SceneVerse++.



(b) An example from ScanNet.

Figure S.1. **Example of quality check.** The data samples from SceneVerse++ and ScanNet are mixed and anonymous.

- **Scene Item Richness**: diversity of abundance of visible items, and how well they reflect realistic indoor layouts.
- **Scene Reconstruction Completeness**: structural completeness of the reconstructed scene, including coherence and absence of major holes or missing regions.
- **Object Reconstruction Completeness**: integrity of individual object shapes, with no breaks, missing faces, or lost components.
- **Object Segmentation Completeness**: whether each object is segmented as a single, coherent instance without obvious omissions or incorrect splits.
- **Object Segmentation Granularity**: the fineness of segmentation, segmenting small objects accurately and avoiding unintended merging.

The results are in Tab. S.1. From the table, SceneVerse++ achieves quality comparable to or exceeding ScanNet across the above evaluation criteria, especially in the richness and completeness of reconstruction, which shows that our dataset captures diverse and real-world distributions. It also indicates that modern image-based reconstruction and segmentation methods, if properly adapted, have advanced to a point where they can surpass the sensor quality and reconstruction pipeline used in ScanNet capture in 2017. This highlights their potential for further scaling. The grading interface is shown in Fig. S.1.

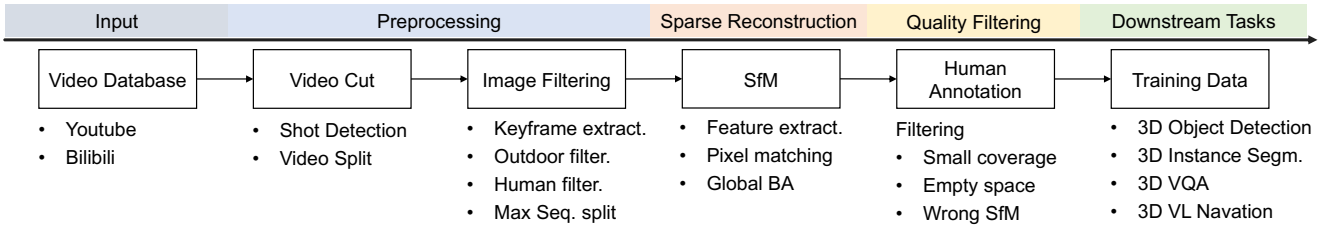


Figure S.2. **Overview of data curation pipeline**.

Table S.1. **The quality check of SceneVerse++ and ScanNet.**

Criterion	SceneVerse++	ScanNet
Scene Item Richness	4.43	3.68
Scene Reconstruction Completeness	4.25	3.09
Object Reconstruction Completeness	4.16	3.23
Object Segmentation Completeness	3.93	3.26
Object Segmentation Granularity	3.89	3.24
Average	4.13	3.30

Table S.2. **Evaluation sensitivity on different segment settings on 3D instance segmentation.** The model is trained with kThresh=10^{-2} and segMinVerts=20, and performance degrades if the distribution of the testing segments diverges from training.

kThresh	segMinVerts	AP_{25}	AP_{50}	AP
10^{-2}	20	36.1	31.8	22.8
10^{-1}	20	34.6	28.1	18.4
10^{-3}	20	35.9	30.4	21.3
10^{-2}	100	30.8	24.6	15.8
10^{-2}	500	17.9	12.6	7.2
10^{-1}	1000	11.2	7.7	4.2
10^{-3}	1000	10.9	7.5	4.1

C. Experiment Details

C.1. 3D Object Detection and Segmentation

3D Object Detection To better handle the large scenes in SceneVerse++, we adopt an additional spatial cropping augmentation during SpatialLM training. For each sample, one object is randomly selected, and the point cloud within a 3-meter radius of the object is extracted and used as the model input. SpatialLM is trained on SceneVerse++ using 8 NVIDIA A100 GPUs for 1000 epochs with a batch size of 1, requiring approximately 2 days. The model is then fine-tuned on ScanNet for another 1000 epochs with a batch size of 4, which takes about 12 hours. For supervision, we utilize 15 semantic categories selected from the ScanNet 20 labels.

3D Instance Segmentation In 3D instance segmentation experiments, we observe that the model trained on SceneVerse++ does not transfer well to ScanNet. One important reason is that Mask3D relies on the segment-level masks produced by a graph-based segmentation method, and different hyperparameters lead to noticeably different segment results. Two decisive hyperparameters, segmentation threshold (kThresh) and minimum segment size (segMinVerts), directly control the connectivity and granularity of segments. To illustrate this sensitivity, we provide further experiments by evaluating a model trained on ScanNet (with kThresh=10^{-2} and segMinVerts=20), on segments generated from different hyperparameter settings. As shown in Tab. S.2 and Fig. S.3, coarse segments fail to correctly isolate individual instances, while overly fine segments result in fragmented predictions and miss detections. This issue is more obvious during the training stage, where the mismatched segment distribution causes poor model transfer. These observations highlight a broader challenge in scaling 3D scene understanding: models sensitive to task-specific modalities and data distribution shifts exhibit limited scalability, whereas models operating directly on raw and widely available modalities may scale more robustly.

C.2. 3D Spatial VQA

Data Generation From the 3D reconstruction and instance segmentation results, we first construct the overall per-scene information, *i.e.*, the room size. Next, we automatically construct 3D scene graphs from point clouds. We first instantiate the graph nodes with the instance annotation from the point cloud and parameterize each node with the object centroid and size of the axis-aligned bounding box. Next, we traverse all the nodes to determine their spatial relationships, following Jia et al. [50]. We then save the counts for different object categories and generate the QAs accordingly.

- Object Count (Numerical Answers (NA)): Count the number of instances of a specified object category that has more than 1 instance within a room.
- Relative Distance (Multiple-Choice Answers (MCA)): Identify which of four candidate objects is closest in 3D space to a target object, which can be uniquely identified by its category.
- Relative Direction (MCA): Given a situation describing the observer's position and orientation, determine the rel-

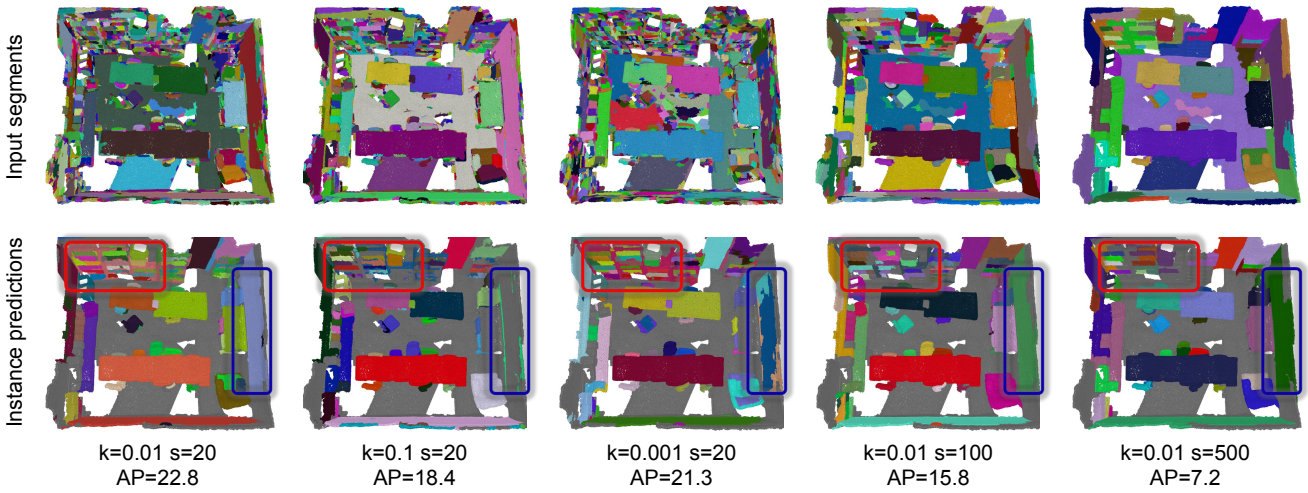


Figure S.3. **Qualitative example of sensitivity tests on different segment distributions.** We evaluate the model trained with k=0.01 and s=20 on other segment settings. The first row shows the input segments, and the second row shows the 3D instance prediction. As the segments become smaller, over-segmentation gradually appears (highlighted by the blue boxes). Conversely, as the segments become larger, under-segmentation becomes increasingly evident (see the red boxes). The AP is reported as the average over the whole ScanNet.

ative direction of a query object, which can be uniquely identified by its category.

- Object Size (NA): Estimate the length of the longest dimension of an object instance in centimeters.
- Absolute Distance (NA): Estimate the Euclidean distance between the closest points of two specified objects in meters. The two objects are randomly selected from the categories that have only one instance.
- Room Size (NA): Estimate the area of the room in square meters (numerical answer).
- Route Planning (MCA): We generate QA pairs by employing the navigation trajectories within 3D environments in the VLN task. The actions are masked to create multiple-choice, and the navigation summary is transferred to guidance via VLM. Detailed prompts are in Tabs. S.7 and S.8.

Dataset Statistics Applying the generation pipeline to SceneVerse++ yields 632K spatial VQA data following the VSI-Bench format. It comprises 391K samples for MCA and 241K samples for NA with 7 different question types. The number of each type of question is listed in Tab. S.3. In our experiment, we sampled a subset of 202K for training.

Training Configuration All experiments for 3D VQA fine-tuning were conducted using LoRA-based adaptation on an LLM backbone, with training performed on 4 × NVIDIA A100 GPUs. More Training Configuration and Reproducibility Details are provided in Tab. S.4.

C.3. 3D Vision-Lanugage Navigation (VLN)

Depth Scale Calibration We design a three-stage pipeline to convert room-tour camera trajectories to VLN trajectories in Sec. 4.3. In action encoding stage, we apply a scale calibration procedure during the action-encoding stage to ensure that movement distances computed from trajectories reflect real-world scale. This is necessary because the SfM reconstruction provides depth on an arbitrary scale, whereas VLN models require physically meaningful forward-motion distances. To estimate the correct scale, we identify video frames containing large and visually stable furniture (e.g., sofas, cabinets, refrigerators), whose depths are easier to estimate reliably. For each selected region, we obtain a robust monocular depth estimate using DepthPro [10]. In parallel, we extract the corresponding absolute (but unscaled) depth from the SfM reconstruction. By comparing these two depth values, we compute a depth-scale factor for each furniture instance. The scale factors are averaged across all selected samples to produce a stable calibration value, which is then applied to the entire reconstructed scene. Accurate depth calibration ensures that

Table S.3. **3D Spatial VQA Data Distribution.**

Task Type	Count
Object Relative Direction	155,199
Object Absolute Distance	137,397
Object Relative Distance	226,639
Object Size Estimate	44,050
Object Count	53,200
Route Plan	9,588
Room Size	6,684
Total	**632,757**

Table S.4. **Training Details for 3D Spatial VQA.**

Category	Setting
Hardware	4 × NVIDIA A100 GPUs
Precision	BF16
LoRA Rank	128
LoRA Scaling Factor	256
Per-device Batch Size	1
Gradient Accumulation Steps	32
Effective Batch Size	$4 \times 32 = 128$
Optimizer	AdamW
Learning Rate	2×10^{-5}
Weight Decay	0
Warmup ratio	0.03
LR Schedule	cosine
Epochs	5
Actual Training	Stop after 1 epoch
Random Seed	42

forward-motion distances derived from trajectories correspond to realistic navigation steps, improving the reliability of action encoding for VLN training. The prompt used for instruction generation is provided in Tab. S.6.

Training Configuration We train LLaVA-Video on 8 NVIDIA A100 GPUs. Zero-shot and mixed-training experiments are run for 1 epoch, while the pretrain–finetune setting uses 2 epochs of pretraining on SceneVerse++ and 1 epoch of fine-tuning on R2R. To ensure balanced exposure to actions across datasets, we apply label rebalancing: we count the occurrences of each action category across all episodes and select a reference frequency based on the median or maximum count. Actions below the reference are oversampled, and actions above the reference are subsampled. Finally, we use the total number of R2R training samples as the baseline and adjust other datasets accordingly to maintain comparable sample counts. Each epoch of training takes approximately 14 hours with a batch size of 2.

Comparison with Internet-Scale VLN Data To validate the effectiveness of our SceneVerse++, we compare it with the YouTube-derived VLN data from NaVILA [24], which contains roughly 20k trajectories, using Qwen-VL-7B [7] as the base model. We evaluate two settings, zero-shot and mixed-training with R2R, and report results in Tab. S.5. In the zero-shot setting, SceneVerse++ and NaVILA show similar performance (SR = 0.09). SceneVerse++ exhibits a larger path length (PL = 11.274), reflecting the inherently longer trajectories present in our data generation pipeline. In the mixed-training setting, SceneVerse++ yields clear improvements over NaVILA on key navigation metrics: higher Success Rate (0.32 vs. 0.29), higher SPL (0.258 vs. 0.213), and lower Distance-to-Goal (7.447 vs. 7.960). This indicates that SceneVerse++ provides more effective supervision for learning grounded navigation when combined with R2R.

Table S.5. **Comparison between NaVILA and SV++.** Experiments use Qwen2.5-VL-7B under zero-shot and mixed-training.

Data Source	Setting	SR↑	OS↑	SPL↑	Dist↓	PL
NaVILA	Zero-shot	**0.09**	0.132	**0.08**	**9.406**	8.505
SV++	Zero-shot	**0.09**	**0.145**	0.063	9.439	11.274
R2R + NaVILA	Mix	0.29	**0.424**	0.213	7.960	16.013
R2R + SV++	Mix	**0.32**	0.402	**0.258**	**7.447**	12.918

Notice that NaVILA contains roughly 2.5× more data than SceneVerse++, which may bias certain metrics in its favor. Despite this scale advantage, SceneVerse++ still achieves superior SR and SPL, suggesting that well-structured, navigation-aligned trajectories are more beneficial than raw data volume alone. These findings support the value of our data-generation pipeline while also underscoring the need to further explore domain differences and dataset scaling in future work.

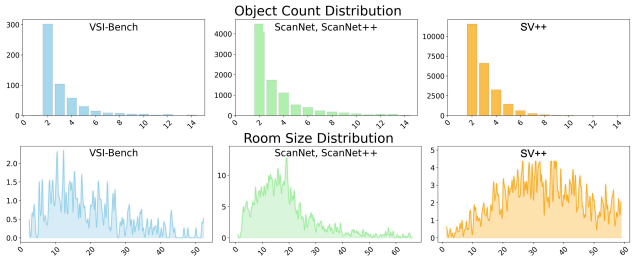


Figure S.4. **3D spatial VQA answer distribution.**

D. More Discussion

Why "Object Count" and "Room Size" performance drop in 3D spatial VQA? We believe the data distribution bias is the major factor here. Several pieces of evidence: 1) SceneVerse++ and ScanNet/ScanNet++ GT perform similarly on zero-shot experiment in Tab. 3; 2) From Fig. S.4, Object Count test distribution in VSI-Bench is highly biased at "2", where in-domain data (ScanNet / ScanNet++) has a much smaller divergence to this peak, showing potential benchmark overfitting:

$$D_{KL}^{obj\_cnt}(\text{VSI-Bench} \parallel \text{SceneVerse++}) = 1.04$$
$$D_{KL}^{obj\_cnt}(\text{VSI-Bench} \parallel \text{SN,SN++}) = 0.145.$$

Room size shows a larger domain gap:

$$D_{KL}^{room\_size}(\text{VSI-Bench} \parallel \text{SceneVerse++}) = 6.08$$
$$D_{KL}^{room\_size}(\text{VSI-Bench} \parallel \text{SN,SN++}) = 2.95,$$

where SceneVerse++ signatures multi-room scenes.

Data scaling analysis We provide scaling results for 3D detection and 3D VQA in Fig. S.5, where data $\sim \mathcal{O}(N_{\text{scenes}})$. Performance follows a *log-linear trend* in both cases, but VQA reaches saturation later. More effective scaling requires co-design involving model architecture, fair benchmarks, and data quality.

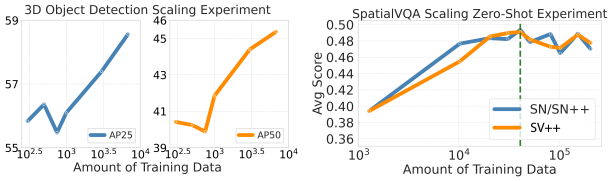


Figure S.5. **Data scaling effects.**

Per-scene computation overhead The average end-to-end per-scene running time is ∼0.59h, consisting of 0.27 GPU-hours (RTX 3090-level) and 0.32h CPU-hour (Xeon 14 vCPUs). Stage-wise, preprocessing and SfM take 69.8%, depth and 2D segmentation model inference takes 23.2%, dense 3D reconstruction 3% and 3D segmentation 4%. This overhead is manageable for large-scale data generation and could be further optimized.

E. Limitations and Future Work

Limited by computational resources, our experiments are bound to the minimal setting to examine the contribution of different data sources. In practice, 3D understanding capability also depends on the base model capacity, optimization strategy, and data mixture, *e.g.*, existing VLN systems often benefit from larger training corpora. Additionally, internet videos may contain privacy-sensitive content from public areas. Scaling such data requires careful adherence to ethical guidelines, regulatory frameworks, and responsible development. Future work includes iterative refinement of the generated data, integration with more advanced models to further enhance capability, and extending to dynamic videos that capture the 4D scene evolution.

Table S.6. **Prompts for Navigation instruction generation in SceneVerse++.**

You are an embodied AI agent making task summaries for a navigation task. Your goal is to generate faithful, human-readable navigation instructions.

— Input —

- A sequence of first-person images and a stepwise action sequence.
- Each image corresponds to the visual observation immediately before the action in that frame.
- Alignment is strictly one-to-one: `image[i]` always pairs with `action[i]`.
- An action entry may describe a single action or a composite action (e.g., "turn left and move forward"), but it still corresponds to the visual state in the paired image.

— Core reasoning rule —

- Always rely primarily on **visual observations** when determining how to move.
- Use actions only as fallback when the image is unclear.
- Maintain consistent spatial logic: if an object is on the left, turning left should bring it to the center view.

— Language and Output Style —

- Avoid first-person narration; use **third-person, objective** instructions such as "A sofa is on the right; turn right to face it."
- Avoid narrative openings (e.g., "The journey begins...").
- Use direct commands: "Turn right into the hallway.", "Walk straight past the sofa.".
- Always include all necessary turning/movement instructions.
- Mention only key orientation-relevant landmarks (sofa, table, doorway, window).

— Responsibilities —

0. Trajectory summarization:
   - Summarize overall motion, room types, and representative objects.
   - Briefly describe the starting location.
   - Provide a concise step-by-step movement description consistent with images.
   - End with a clear final position description.

1. Per-step reasoning:
   - Think in first-person as the agent (camera aligned with orientation).
   - Base reasoning on **visible evidence** in the current frame.
   - Mention only representative, orientation-relevant objects.
   - Use diverse spatial expressions: "to the right", "just ahead", "past the table", etc.
   - Ensure geometric consistency between viewpoint and actions.
   - If actions conflict with geometry, trust the image.

— Action rules —

- Actions may be single or composite (joined by "and").
- Allowed actions: `TurnLeft`, `TurnRight`, `MoveForward`, `Move`, `Stop`.
- "Move" alone means a small forward motion without rotation.
- Composite actions operate sequentially: turn first, then move.

— Special Requirement: Three Reformulations —

Rewrite the trajectory summary into **three distinct linguistic styles** with identical semantic content: Formal Instructional Style, Natural Conversational Style and Narrative Descriptive Style.

Guidelines:

- All three must preserve identical spatial logic and landmarks.
- No conflicts are allowed.
- All must fully cover the entire trajectory.

— Examples of the Three Styles —

- **Instruction 1 (Formal):** "Turn right into the hallway. Advance straight past the dining table. Enter the bedroom and stop in front of the bed."
- **Instruction 2 (Conversational):** "Take a right into the hallway and keep walking until you pass the dining table on your left. Go into the bedroom and stop by the bed."
- **Instruction 3 (Narrative):** "Turning right, you move into the hallway, the dining table sliding by on your left. The hall opens into a bedroom, where you halt just before the bed."

Table S.7. **Prompts to generate route plan VQA in SceneVerse++- part1.**

You are an AI assistant tasked with generating **Fill-in-the-blank Action Completion MCQ** for robot navigation. Your job is to output a multiple-choice question (with blanks) and its correct answer.

— Input —

A sequence of continuous key frames from a room-tour video (the frames are consecutive and represent a smooth camera/robot trajectory).

— High-level rule (priority order) —

1. ALL reasoning must be grounded purely on the **visual evidence from the frames**
2. Use visual cues such as object appearance/disappearance, relative positions, scaling, and viewpoint rotation to infer the robot's movements and turns.
3. When describing places, objects, or targets, use detailed and specific visual anchors — not just generic room names. For example: "the blue sofa on the right," "the black dining table ahead," "the kitchen counter with sink," or "the hallway with a white door at the end."
4. If any step or turn cannot be confidently inferred from visual evidence, skip or merge it rather than guessing. Do NOT fabricate movements.

— Core Procedure (must follow this order) —

1. **Construct a concise, numbered Trajectory Summary**:
   - Carefully analyze the continuous frame sequence to extract a minimal yet complete trajectory.
   - Each entry in the summary should be a single action step, such as: "1. Go forward until [object/room]", "2. Turn left", "3. Go forward until [object/room]".
   - Determine steps by tracking:
     - Appearance/disappearance or scaling of landmarks (for "Go forward")
     - Change in viewing direction or lateral movement (for "Turn left/right/back")
   - The summary should form a coherent navigation sequence from the starting viewpoint to the final destination.
   - Explicitly describe both start place and end place in visually grounded detail: e.g., "You are a robot beginning at the living room, facing the blue sofa." e.g., "You want to navigate to the kitchen with a table on your left."
   - When describing each "Go forward" anchor, be as specific as visually supported:
     - Include object appearance (color, size, material), or scene context (e.g., furniture type or nearby area). Example: "Go forward until the blue sofa."
   - Only include meaningful transitions — skip redundant minor movements or rotations that don't correspond to clear spatial change.
   - Ensure geometric reasoning consistency.
   - Make the trajectory alternate logically between "Go forward" and "Turn".
   - Example of a trajectory summary:
     - 1. Go forward until the 3-seater sofa (evidence: frames X–Y)
     - 2. Turn right (evidence: frames X–Y)
     - 3. Go forward until the dining table (evidence: frames X–Y)
     - 4. Turn left (evidence: frames X–Y)
     - 5. Go forward until the kitchen counter with sink (evidence: frames X–Y)
2. **QA Generation (based on Trajectory Summary only)**:
   - Normalize steps to alternate between "Go forward…" and "Turn …".
   - Determine where to place `[please fill in]` blanks:
     - Every turn step must become a blank.
   - "Go forward" must mention detailed visible landmarks.
   - Use the strict template:

```
Q: You are a robot beginning at [start place, with visual details].
You want to navigate to [end place, with visual details].
You will perform the following actions:
1. Go forward until [object/room]
2. [please fill in]
3. Go forward until [object/room]
...
N. Go forward until [object/room].
You have reached the final destination.
(Note: for each [please fill in], choose either
'turn back,' 'turn left,' or 'turn right.')
```

Table S.8. **Prompts to generate route plan VQA in SceneVerse++- part2**

3. Options generation:
- For each blank, permissible options:
  - 'turn back', 'turn left', 'turn right'
- If one blank: produce A–C.
- If >= 2 blanks: produce A–D.
- Each option is a full sequence of turns.

4. Correct answer determination:
- Must match turns inferred from visual trajectory summary.
- No guessing ambiguous turns.

— Output format (STRICT JSON SCHEMA — all keys required) —
- "trajectory summary": a list of strings.
- "question": full question string.
- "options": dictionary with keys A, B, C, D.
- "answer": one correct option.

— Strict behaviour notes (must obey) —
- Only use frame-based evidence.
- Build a reliable Trajectory Summary.
- The first step may be a Turn or a Go-forward action.
- Never guess ambiguous turns.