Context-Nav: Context-Driven Exploration and Viewpoint-Aware 3D Spatial Reasoning for Instance Navigation

Supplementary Material

A. Overview

In this supplementary material, we provide additional details and analyses that complement the main paper:

- **Sec. B (Related Work)** summarizes additional background on visual navigation goal formulations and zero-shot ObjectNav pipelines.
- **Sec. C (Text-Goal Preprocessing)** details our LLM-based pipeline that decomposes free-form text goals into intrinsic attributes, context objects, and spatial-relation triples used for downstream reasoning.
- **Sec. D (Viewpoint-Aware 3D Verification of Extrinsic Attributes)** explains how viewpoint-aware 3D spatial reasoning is performed to verify extrinsic contextual relations between the target and surrounding objects.
- **Sec. E (Prompts)** lists the exact LLM and VLM prompts used for attribute extraction, question generation, context filtering, relation extraction, non-COCO instance verification, and intrinsic attribute scoring.
- **Sec. F (Additional Experimental Results)** reports further quantitative results, including a re-evaluation of InstanceNav under stricter success criteria and an analysis of Context-Nav's performance.
- **Sec. G (Implementation Details)** provides implementation specifics such as model choices, agent configuration, and hyperparameters for detection, mapping, and wall reconstruction, and additionally reports runtime/latency statistics.
- **Sec. H (Category Coverage on CoIN-Bench)** summarizes the target category coverage of CoIN-Bench and the resulting diversity of the evaluated open-vocabulary category space.
- **Sec. I (Failure Case Analysis)** analyzes failure episodes (time-out or false-positive stop) and discusses dominant failure modes, including planning under imperfect geometry, perception/detection failures, intrinsic ambiguity, and room-segmentation errors.
- **Sec. J (Qualitative Comparison with RL-Trained and Training-Free Baselines)** presents qualitative visualizations comparing Context-Nav with RL-trained and training-free baselines on InstanceNav and CoIN-Bench.

B. Related Work

Goal formulations in visual navigation differ primarily in how the target is specified and how strongly the trajectory is constrained. For completeness, we expand on the task variants summarized in the main paper. Point-goal navigation [5] specifies the target as a metric coordinate in the environment, and the agent must reach this location using onboard sensing and odometry. Image-goal navigation instead provides a reference image that the agent must match by exploring the scene [15, 28]. Category-level ObjectNav asks the agent to reach any instance of a named class (e.g., "a chair") [5, 20], while vision-and-language navigation extends this to stepwise natural-language instructions or route descriptions that constrain the path the agent should follow [2, 3]. These formulations differ in how precisely they specify the final instance and path, and each introduces characteristic limitations: category-level goals under-specify which instance should be selected, image-goal settings presuppose access to an ideal goal snapshot, and instruction-following tasks assume detailed human-authored guidance at test time.

Zero-shot ObjectNav methods relax the fixed task taxonomy by enabling open-vocabulary goal specifications at inference time. A common strategy replaces static category embeddings with CLIP-style text–image encoders or retrieval modules that map free-form textual queries to visual exemplars [1, 13, 16]. Training-free modular pipelines further use vision–language cues as an exploration prior: frontier candidates on a top-down occupancy map are ranked using CLIP similarities [8] or text-conditioned value maps [17, 23], and large language or vision–language models score these frontiers based on detected-object lists or semantic maps before a classical planner executes the selected action [6, 10, 12, 19, 21, 24–26]. While these approaches already exploit language as a high-level prior over frontiers, they typically operate at the category level or with short attribute snippets, and do not yet elevate long, contextual descriptions to the central exploration and verification signal that Context-Nav provides.

C. Text-Goal Preprocessing

C.1. Benchmark-Specific Goal Decomposition

Text-goal decomposition follows a shared pipeline across benchmarks, with minor task-specific differences for InstanceNav [16] and CoIN-Bench [17]. On InstanceNav, the benchmark already provides a target category together with two separate strings: an intrinsic-attribute description and an extrinsic/context description. These two strings are directly treated as the intrinsic part and the context part of G. On CoIN-Bench, the benchmark instead provides a single caption that mixes intrinsic and extrinsic cues; this full cap-

tion is used both as the input to intrinsic-attribute extraction and as the context-description text for context objects and relations. In both cases, the target category label is supplied to all prompts as metadata.

C.2. Intrinsic Attributes and Attribute Questions

Intrinsic attributes of the target are extracted by a single LLM call conditioned on the linguistic description and the target category. Using the intrinsic-attribute extraction prompt (Sec. E), the model is instructed to return only attributes that explicitly modify the target category in G, restricted to color and shape; attributes of other objects in the caption are ignored. The output is a small key–value dictionary (e.g., `color: "yellow and green"`, `shape: "square"`).

The resulting attribute dictionary is then converted into a set of binary questions used for yes/unknown/no intrinsic verification. Through the attribute question generation prompt, the LLM receives the target category and the extracted attributes and returns a set of concise, unambiguous yes/no questions for each attribute type (e.g., "Is the outlined picture mainly yellow and green?"). These questions are later issued to the VLM when a candidate instance is observed during the intrinsic-verification stage in Sec. 3.4 of the main paper.

C.3. Context Objects and Spatial Relations

Context-object categories are derived by filtering and canonicalizing noun phrases extracted from the context part of G. First, candidate noun phrases are extracted with a syntactic parser [9] and lightly normalized (e.g., "the wooden cabinet"). Second, the normalized list is passed to the context object filtering prompt, which retains only phrases that refer to concrete physical indoor objects and discards directions, regions, and purely material/color/shape expressions. Third, using the context synonym grouping prompt, the LLM groups the remaining phrases into synonym clusters, assigns a canonical label to each group (aligned with common indoor categories, e.g., "cabinet"), and enforces that any group containing the target category uses the exact target string as its canonical key. This procedure yields a set of canonical context categories together with a mapping from raw phrases to canonical labels that is reused for relation extraction.

Context descriptions are subsequently converted into symbolic spatial relations over the allowed object terms. Using the spatial relation extraction prompt, the LLM receives the full caption and the list of allowed raw terms (all canonical context categories plus the target category) and is instructed to output a small set of unambiguous pairwise relations $(\text{ref}, \text{tgt}, \rho)$, where $\rho \in \{\texttt{left}, \texttt{right}, \texttt{front}, \texttt{behind}, \texttt{near}, \texttt{above}, \texttt{below}\}$. The prompt constrains the model to use only allowed surface forms for objects and to skip ambiguous pairs; the raw object names are then mapped back to canonical categories. The resulting relation set $\mathcal{T}$ and the associated instance centers form the input to the viewpoint-aware 3D spatial reasoning stage in Sec. 3.4 of the main paper.

D. Viewpoint-Aware 3D Verification of Extrinsic Attributes

Viewpoint-aware 3D verification operationalizes extrinsic attributes by testing spatial-relation triples against the reconstructed instance-level map under viewpoint uncertainty. Given the goal G, the extrinsic/context part of the description is parsed into spatial-relation triples $(\text{ref}, \text{tgt}, \rho)$, and candidate context instances and relation sets are derived as in Sec. C. As the agent explores, the perception and mapping modules build instance-level 3D point clouds and a wall-only map that defines wall-bounded rooms. After room-level filtering ensures that the candidate target instance and at least one context instance are co-located in the same wall-bounded room (Step 1 in Sec. 3.4 of the main paper), the pipeline samples the candidate viewpoint set $\mathcal{V}$ around the reference–target pairs, aligns a local frame at each viewpoint, and evaluates the spatial predicates defined in Eq. (5) of the main paper. Figure S1 provides a visual overview of this procedure.

E. Prompts

This section lists the LLM and VLM instructions used in the text-goal preprocessing and verification pipeline. In all prompts, tokens written in `UPPERCASE` denote placeholders that are replaced at runtime by episode-specific values (e.g., the target category or the goal caption).

E.1. Intrinsic Attribute Extraction

In the intrinsic-attribute extraction prompt, {`TARGET`} is replaced with the target category (e.g., "picture") and {`TEXT`} with either the intrinsic-attribute string (InstanceNav) or the full caption (CoIN-Bench). The model is expected to return a JSON-style dictionary whose keys are a subset of `"color"` and `"shape"` and whose values are free-form textual descriptions.

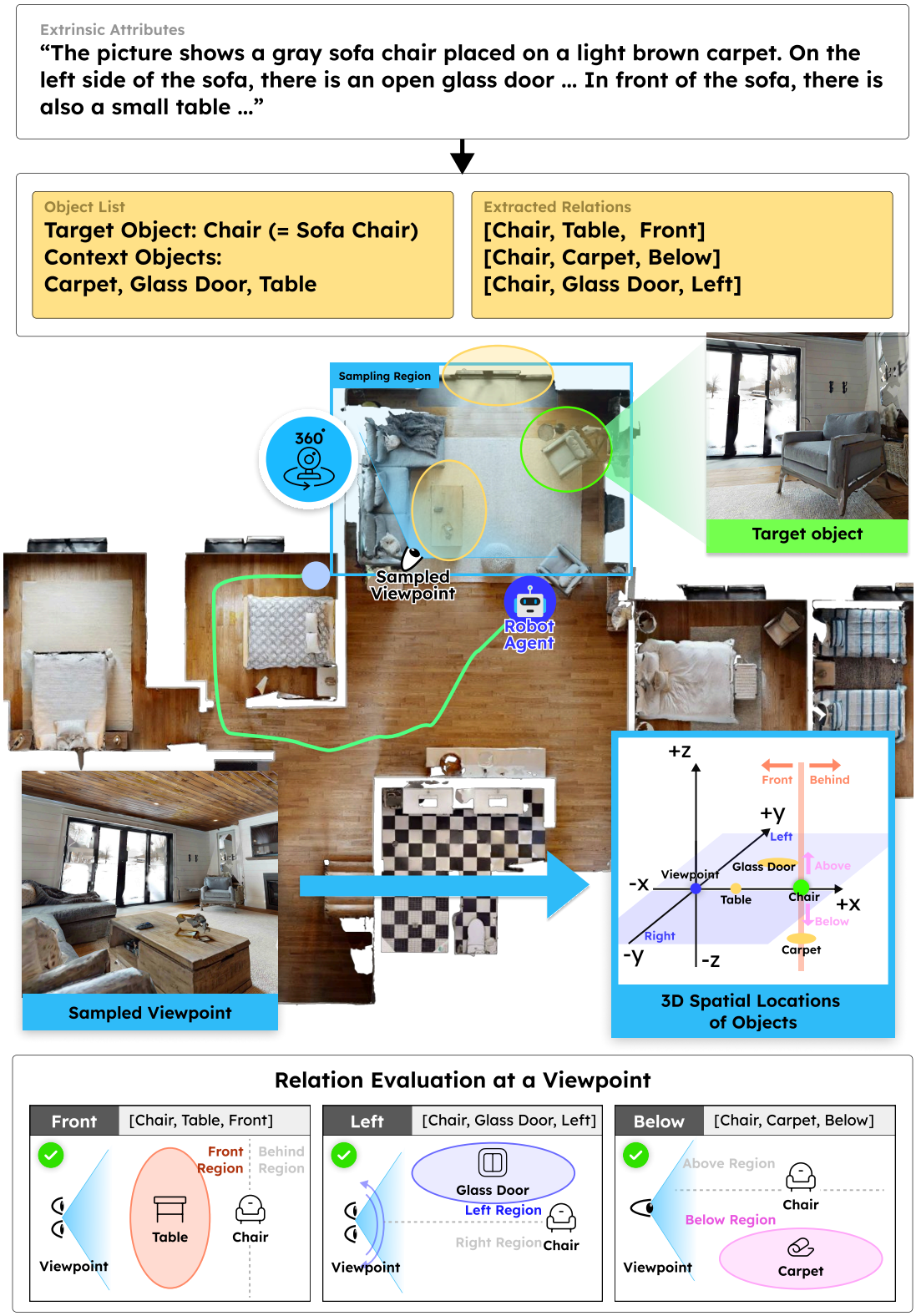

Figure S1. **Viewpoint-aware 3D verification of extrinsic attributes.** Starting from the extrinsic part of the goal description, Context-Nav extracts context objects and spatial-relation triples, builds instance-level 3D point clouds, and samples candidate viewpoints around the reference–target pairs. For each candidate viewpoint, a local frame is aligned and the corresponding spatial predicates are evaluated; the figure visualizes an example viewpoint from which all extrinsic relations are satisfied simultaneously, together with the resulting relation checks for the confirmed target instance.

You extract visual attributes from a short object description.
INPUT: a TARGET category {TARGET} and a description {TEXT}.
1. Read the description and identify attributes that explicitly modify the TARGET object.
2. Only consider the following attribute types: color and shape.
3. Ignore attributes of any other object, even if they are in the description.
4. Do not infer or guess; only use attributes that are clearly stated.
OUTPUT: a JSON dictionary with up to two keys:
{ "color": string or null, "shape": string or null }
Omit any key whose attribute is not explicitly mentioned.

E.2. Attribute Question Generation

For attribute question generation, {TARGET} is replaced with the target category and {ATTRIBUTES} with the JSON dictionary returned by the previous prompt. The model outputs a list of question–attribute-type pairs.

You convert attributes into direct yes/no VQA questions for a vision-language model.
INPUT:
- TARGET category: {TARGET}
- Attribute dictionary: {ATTRIBUTES}, a JSON with keys from ["color", "shape"].
1. For each attribute present in {ATTRIBUTES}, write concise, unambiguous yes/no questions about a single instance of the TARGET.
2. Each question must explicitly mention the TARGET and the attribute value (e.g., color phrase).
3. Do not create questions for attributes that are missing from the dictionary.
4. Do not ask about other objects or unstated properties.
OUTPUT: a JSON list of objects of the form
[{"atype": "color", "q": "...?"}, {"atype": "color", "q": "...?"}, {"atype": "shape", "q": "...?"}, ...]
where atype is either "color" or "shape".

E.3. Context Object Filtering

For context object filtering, {PHRASES} is replaced with the list of normalized noun phrases extracted from the context description (one phrase per line or as a JSON list).

You are given candidate noun phrases from an indoor scene description.
INPUT: a list of noun phrases {PHRASES}.
1. Decide which phrases refer to concrete physical objects that can plausibly exist indoors (e.g., "cabinet", "staircase", "pillow").
2. EXCLUDE phrases that are: - pure directions or regions (e.g., "left side", "corner of the room"), - purely generic items without a concrete object (e.g., "stuff", "some things"), - phrases that only name materials, colors, or shapes (e.g., "wooden", "red", "round object" without a category).
3. Do not invent new words or rewrite phrases; you may only select from the input list.
OUTPUT: a JSON list of the selected phrases, preserving their original text.

E.4. Context Synonym Grouping

For context synonym grouping, {TERMS} is replaced with the filtered phrases from the previous step and {TARGET} with the target category.

You group indoor object names into synonym clusters and assign a canonical label to each cluster.
INPUT:
- Filtered phrases: {TERMS}
- Target category: {TARGET}
1. Treat each item in {TERMS} as a candidate object name.
2. Group terms that refer to the same indoor household object category (e.g., "sofa" and "couch").
3. For each group, choose a canonical label:
- Use a simple, lowercase, space-separated form (no underscores).
- If any term in the group exactly matches {TARGET}, then set the canonical label of that group to {TARGET} exactly.
4. Do not introduce new object names; every term in each group must come from {TERMS}.
OUTPUT: a JSON object
{ "canonical_1": ["term_a", "term_b", ...], "canonical_2": ["term_c", ...], ... }
covering all grouped terms.

E.5. Spatial Relation Extraction

For spatial relation extraction, {CAPTION} is replaced with the full goal description G and {ALLOWED_TERMS} with the list of allowed raw object names (target category plus all canonical context terms and their raw variants).

You convert a single-view image caption into pairwise spatial relations among indoor objects.
INPUT:
- Caption describing an indoor scene: {CAPTION}
- Allowed object names (surface forms): {ALLOWED_TERMS}
1. Identify all mentions of objects whose names (or synonyms) appear in {ALLOWED_TERMS}.
2. From the caption, extract pairwise spatial relations of the form (REFERENCE object, TARGET object, RELATION), where RELATION is one of:
- "left", "right", "front", "behind", "near", "above", "below".
3. The caption may describe frame-relative positions (e.g., "left side of the image", "upper right corner"). Interpret them in terms of 2D image coordinates only (no depth reasoning), e.g.:
- Object on the left/upper-left/lower-left of the frame is LEFT of an object on the right/upper-right/lower-right.
- Object in the lower/bottom region is BELOW an object in the upper/top region.
4. Use ONLY the provided {ALLOWED_TERMS} as surface forms for "ref" and "tgt". If the text uses a synonym not in the list, map it to the nearest allowed label and output the allowed label.
5. Emit only relations that are clearly entailed by the caption. Skip ambiguous, underspecified, or contradictory relations (e.g., both left(A,B) and left(B,A)).
6. Prefer a concise set of relations: output at most 6 of the most salient pairs.
OUTPUT: a JSON list of relations, each of the form { "ref": "object_1", "tgt": "object_2", "rtype": "relation", ... } where `rtype` is one of ["left","right","front","behind","near","above","below"].

E.6. Non-COCO Instance Verification

Open-set categories that are not covered by COCO are verified by prompting the VLM to decide whether a mask-highlighted region belongs to a proposed category. The placeholder {CATEGORY_NAME} is filled with the detector's open-vocabulary label (e.g., "dresser", "radiator"), and the RGB frame is annotated with an outline around the candidate region before being passed to the VLM. The returned `prob` is interpreted as a calibrated confidence in $[0, 1]$, and instances are treated as valid when $\texttt{prob} \geq 0.6$.

Is the outlined object an instance of the category "{CATEGORY_NAME}"? Judge ONLY the pixels inside the outline/mask; ignore everything else. If the outline is faint, assume the highlighted/outlined region marks a SINGLE candidate instance.
Return ONLY JSON: {"prob": float between 0 and 1} where prob is P(instance $\in$ category). If the region is too small/blurred/occluded or ambiguous, return a value near 0.5.

E.7. Attribute Scoring for Intrinsic Verification

For intrinsic attribute verification, the VLM is instructed to act as a strict visual judge that scores whether a single attribute claim about the highlighted instance is true, using only the given image. The placeholder {ATTRIBUTE_QUESTION} is filled with a paraphrased yes/unknown/no-style question produced by the LLM, and the resulting integer score in $\{0, \ldots, 15\}$ is mapped to the Yes/Unknown/No bins described in Sec. 3.4 of the main paper:

You are a strict visual judge. Score whether the claim about the object is true using ONLY the provided image. SCORING (integer 0..15):
- NO [0..4]
- YES [11..15]
- UNKNOWN [5..10] (visibility limits only).
Never output text other than the JSON below.
CLAIM: Question about the outlined object: {ATTRIBUTE_QUESTION}
Return ONLY valid JSON: {"score": integer 0..15}

F. Additional Experimental Results

Method	Model Condition		InstanceNav	
	Input	Training-free	SR↑	SPL↑
PSL [16]	d	✗	18.0	7.2
VLFM [23]	c	✓	9.5	5.7
UniGoal [22]	d	✓	13.5	6.7
Ours	d	✓	**21.2**	**7.9**

Table S1. **Benchmark results on InstanceNav.** Comparison of RL-trained policies, training-free modular baselines, and the proposed Context-Nav on InstanceNav [16] under the stricter CoIN-Bench [17] success criteria. Input type c denotes a category-level goal specification, while d denotes a language description of the target.

Additional experiments re-evaluating InstanceNav [16] under the stricter CoIN-Bench [17] success criteria show that Context-Nav maintains state-of-the-art SR and SPL among both RL-trained and training-free methods. As summarized in Table S1, CoIN-Bench reduces the success radius from 1m to 0.25m and shortens the episode horizon from 1,000 to 500 steps, forcing agents to stop precisely at the described instance among same-category distractors while exploring efficiently. Under this strict setting, Context-Nav achieves 21.2% SR and 7.9 SPL, indicating that leveraging contextual information with viewpoint-aware 3D reasoning remains effective even when approximate stops and long trajectories are strongly penalized.

G. Implementation Details

G.1. Language and Vision-Language Models

Language and vision–language components of Context-Nav are instantiated with GPT-OSS 20B [14] and Qwen2.5-VL 7B [4], respectively, for all experiments. GPT-OSS 20B parses the free-form goal G into intrinsic and extrinsic attributes, extracts relation triples $(\text{ref}, \text{tgt}, \rho)$, and generates paraphrased yes/unknown/no question templates for attribute verification (Sec. C). Qwen2.5-VL 7B is employed both as the open-set category verifier in Sec. 3.2 of the main paper (for non-COCO categories that require yes/no classification on masked regions) and as the intrinsic-attribute VQA module in Sec. 3.4, where it answers attribute-specific questions on the highlighted object and returns discretized confidence scores $s \in \{0, \ldots, 15\}$. Each model is used with its default inference hyperparameters (temperature, top-p), and no fine-tuning or in-domain adaptation is performed.

G.2. Agent Configuration

Agent embodiment and action space are kept identical across InstanceNav and CoIN-Bench. In both benchmarks, the agent uses the discrete action set `forward`, `turn-left`, `turn-right`, `stop` as defined in Sec. 3.1 of the main paper. A `forward` action moves the agent by 0.25m along its current heading, while `turn-left` and `turn-right` rotate the agent by 30°.

G.3. Hyperparameters

Detection thresholds. Detection thresholds in the open-vocabulary detection and verification module (Sec. 3.2, "Open-Vocabulary Detection and Verification") are configured as follows. GroundingDINO [11] proposals are accepted when their confidence is at least 0.45, YOLOv7 [18] detections are accepted when the confidence is at least 0.8, and VLM yes/no classification outputs are promoted to persistent instances when their score in $[0, 1]$ is greater than or equal to 0.6.

Spatial proximity heuristic. The spatial proximity heuristic for first-stage association in the instance-level 3D mapping module (Sec. 3.2, "Instance-Level 3D Mapping") merges two observations when the Euclidean distance between their 2D centers on the ground plane is below 0.26m, thereby cheaply capturing short-term revisits.

Voxel-overlap association. Voxel-overlap association in Eq. (1) of the main paper relies on discretizing each instance point cloud into a voxel grid with resolution 0.05m. Up to 5,000 points are uniformly sampled per instance (or all points if fewer are available), and the overlap score $s(A, B)$ is computed on the resulting voxel sets. Two instances are merged when $s(A, B) > 0.45$; otherwise, a new instance is created.

Wall-map RANSAC. RANSAC parameters for constructing the wall-only map (Sec. 3.2, "Wall-Only Map") are chosen to robustly recover structural planes. Depth points are range-gated to $[0.5, 5]$ meters before RANSAC [7, 27]; a 0.03m inlier distance threshold is used, at least 400 inliers per plane are required, and the maximum number of iterations is capped at 1,500. A vertical-plane constraint on the fitted normal ($n_z \leq 0.3$) is additionally enforced, and at most three structural planes per frame are retained when updating the wall-only layer.

G.4. Runtime and Latency

Component	Det/Seg Modules	Value-map Update	3D Association	VLM Inference	Viewpoint-aware Verification
Context-Nav	0.12	0.11	0.29	0.49	0.0004

Table S2. **Per-call latency (in seconds) of each module.**

On a single NVIDIA A100 GPU, Context-Nav runs at 0.54 s/step on average (range: 0.36–1.12 s/step). Table S2 summarizes the per-call latency of each component. Crucially, VLM inference is not executed at every step: it is triggered in only 10% of steps (8% for open-set category verification and 2% for intrinsic-attribute verification), and most steps incur no VLM overhead. For reference, representative modular baselines are substantially slower:

AIUTA runs at 1.8 s/step on average (0.9–7.9 s/step), and UniGoal runs at 0.8 s/step (0.4–12 s/step).

H. Category Coverage on CoIN-Bench

Category coverage on CoIN-Bench [17] is broader and more diverse than on InstanceNav [16] while still overlapping with the six InstanceNav categories (chair, sofa, TV, bed, toilet, and plant). Across both benchmarks, the evaluation therefore includes common InstanceNav targets such as chair, bed, plant, and TV as well as a wide range of additional furniture, storage, textile, and decorative objects, allowing us to test whether context-driven exploration and 3D spatial verification generalize beyond a small, fixed category set.

The CoIN-Bench splits further diversify the target space. On `Val Seen`, the agent is evaluated on categories including cabinet, bed, table, clothes, kitchen lower cabinet, blanket, cloth, TV, bathroom cabinet, wardrobe, desk, display cabinet, chair, heater, rack, and board. The `Val Unseen` split introduces additional target categories such as mirror, picture, dresser, radiator, plant, window glass, hanging clothes, rug, and book. Finally, the `Val Seen Synonyms` split includes kitchen cabinet, armchair, and shelf, and deliberately tests robustness to lexical variation (e.g., armchair vs. chair) and fine-grained category distinctions. This combination of overlapping InstanceNav categories and split-specific CoIN-Bench targets ensures that the reported SR and SPL reflect performance across a heterogeneous, open-vocabulary category space rather than a narrow, benchmark-specific label set.

I. Failure Case Analysis

We define a failure episode as either (i) time-out, where the agent exceeds the maximum step budget without reaching the described instance, or (ii) a false-positive stop, where the agent terminates at a distractor of the same category or an off-target object. We group failures into three dominant modes.

Planning. A common failure source is planning under imperfect geometry. In some scenes, phantom free-space artifacts in HM3D lead the global planner to produce paths that are not executable in practice. In addition, the agent may fail to perceive obstacles early enough due to its limited field of view, which can cause late replanning and unrecoverable detours near the end of an episode.

Detection. Failures in the perception stack typically arise from noisy or missing open-vocabulary detections. When either the target or key context objects are missed (or spuriously detected), downstream mapping and verification become unreliable. A related issue is limited observability: some context objects required by the description are not visible from the viewpoints encountered before the episode ends, preventing the pipeline from accumulating sufficient evidence for verification.

Ambiguity. Some episodes are intrinsically ambiguous: multiple same-category instances remain consistent with the text description under partial observations, and the caption alone does not provide enough constraints to uniquely identify the intended instance. In such cases, additional sensing actions (e.g., deliberate viewpoint changes) or external interaction would be required to guarantee success.

J. Qualitative Comparison with RL-Trained and Training-Free Baselines

Qualitative comparisons with RL-trained and training-free baselines on CoIN-Bench [17] and InstanceNav [16] complement the quantitative results by visualizing how agents behave in cluttered scenes (Figs. S2 and S3). For each dataset, the RL-trained PSL [16] policy serves as the training-based baseline, and a representative training-free pipeline is chosen (AIUTA [17] on CoIN-Bench and UniGoal [22] on InstanceNav). Each panel visualizes a top-down map together with the natural-language goal, a crop of the target object, and the navigation trajectories of all methods. The trajectory of Context-Nav is drawn in orange, while trajectories of the baseline methods are shown in light gray. The final stopping location of each agent is annotated as *Target* when it reaches the correct instance, *Time-out* when the maximum step budget is exceeded without reaching the goal, *Distractor* when the agent stops at a different instance of the same category, and *Off-target* when it stops at an object from a different category. Across both benchmarks, the illustrated episodes show that Context-Nav more reliably reaches the correct target and tends to follow shorter, context-consistent routes, whereas the baselines often time out or terminate at distractors or off-target objects.

J.1. CoIN-Bench

Figure S2. **Qualitative comparison on CoIN-Bench.** Context-Nav trajectories are compared with those of the RL-trained PSL [16] policy and the training-free AIUTA [17] agent on CoIN-Bench episodes featuring multiple same-category distractors. For each text goal, top-down trajectories are overlaid on the floor map: Context-Nav is shown in orange and the baselines in light gray. Insets show the final egocentric view and outcome label for each method: *Target* indicates that the correct instance is reached within the step budget, *Time-out* denotes failure to reach any candidate in time, and *Distractor* indicates that the agent stops at a different instance of the same category. In these examples, Context-Nav consistently reaches the correct instance, whereas the baselines either time out or stop at distractors.

J.2. InstanceNav

Figure S3. **Qualitative comparison on InstanceNav.** Representative episodes from InstanceNav [16] compare Context-Nav with the RL-trained PSL [16] policy and the training-free UniGoal [22] pipeline. As in Fig. S2, top-down trajectories and final goal views are visualized, drawing Context-Nav in orange and the baselines in light gray. The terminal state is annotated as *Target* when the correct instance is reached, *Distractor* when the agent stops at a different instance of the same category, and *Off-target* when the agent stops at an object from a different category than the described target. These examples highlight that Context-Nav successfully reaches the correct goal instances, whereas PSL and UniGoal often time out, stop at same-category distractors, or end up off-target at objects of incorrect categories.

References

[1] Ziad Al-Halah, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Zero experience required: Plug & play modular transfer learning for semantic visual navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 17031–17041, 2022. 1

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3674–3683, 2018. 1

[3] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3674–3683, 2018. 1

[4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. arXiv preprint arXiv:2502.13923, 2025. 6

[5] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. ArXiv, abs/2006.13171, 2020. 1

[6] Wenzhe Cai, Siyuan Huang, Guangran Cheng, Yuxing Long, Peng Gao, Changyin Sun, and Hao Dong. Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill. In 2024 IEEE International Conference on Robotics and Automation, pages 5228–5234. IEEE, 2024. 1

[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981. 6

[8] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 23171–23181, 2023. 1

[9] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In Conference on Empirical Methods in Natural Language Processing, 2015. 2

[10] Yuxuan Kuang, Hai Lin, and Meng Jiang. OpenFMNav: Towards open-set zero-shot object navigation via vision-language foundation models. In Findings of the Association for Computational Linguistics: NAACL 2024, pages 338–351, Mexico City, Mexico, 2024. Association for Computational Linguistics. 1

[11] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In European Conference on Computer Vision, pages 38–55. Springer, 2024. 6

[12] Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. In Proceedings of The 8th Conference on Robot Learning, pages 2049–2060. PMLR, 2025. 1

[13] Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. Advances in Neural Information Processing Systems, 35:32340–32352, 2022. 1

[14] OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. 6

[15] Xinyu Sun, Peihao Chen, Jugang Fan, Thomas H. Li, Jian Chen, and Mingkui Tan. Fgprompt: Fine-grained goal prompting for image-goal navigation. ArXiv, abs/2310.07473, 2023. 1

[16] Xinyu Sun, Lizhao Liu, Hongyan Zhi, Ronghe Qiu, and Junwei Liang. Prioritized semantic learning for zero-shot instance navigation. In European Conference on Computer Vision, pages 161–178. Springer, 2024. 1, 6, 7, 8, 9

[17] Francesco Taioli, Edoardo Zorzi, Gianni Franchi, Alberto Castellini, Alessandro Farinelli, Marco Cristani, and Yiming Wang. Collaborative Instance Object Navigation: Leveraging Uncertainty-Awareness to Minimize Human-Agent Dialogues. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 18781–18792, 2025. 1, 6, 7, 8

[18] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7464–7475, 2023. 6

[19] Pengying Wu, Yao Mu, Bingxian Wu, Yi Hou, Ji Ma, Shanghang Zhang, and Chang Liu. Voronav: Voronoi-based zero-shot object navigation with large language model. In International Conference on Machine Learning, pages 53757–53775. PMLR, 2024. 1

[20] Karmesh Yadav, Jacob Krantz, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Jimmy Yang, Austin Wang, John Turner, Aaron Gokaslan, Vincent-Pierre Berges, Roozbeh Mootaghi, Oleksandr Maksymets, Angel X Chang, Manolis Savva, Alexander Clegg, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2023. https://aihabitat.org/challenge/2023/, 2023. 1

[21] Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. Advances in Neural Information Processing Systems, 37:5285–5307, 2024. 1

[22] Hang Yin, Xiuwei Xu, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. Unigoal: Towards universal zero-shot goal-oriented navigation. In Proceedings of the Computer Vision

and Pattern Recognition Conference, pages 19057–19066, 2025. 6, 7, 9

[23] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In 2024 IEEE International Conference on Robotics and Automation, pages 42–48. IEEE, 2024. 1, 6

[24] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3mvn: Leveraging large language models for visual target navigation. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3554—3560. IEEE, 2023. 1

[25] Lingfeng Zhang, Qiang Zhang, Hao Wang, Erjia Xiao, Zixuan Jiang, Honglei Chen, and Renjing Xu. Trihelper: Zero-shot object navigation with dynamic assistance. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 10035–10042. IEEE, 2024.

[26] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In International Conference on Machine Learning, pages 42829–42842. PMLR, 2023. 1

[27] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. arXiv:1801.09847, 2018. 6

[28] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In 2017 IEEE International Conference on Robotics and Automation, pages 3357–3364. IEEE, 2017. 1