

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 CoNAV BENCH: COLLABORATIVE LONG-HORIZON VISION-LANGUAGE NAVIGATION BENCHMARK

Anonymous authors

Paper under double-blind review

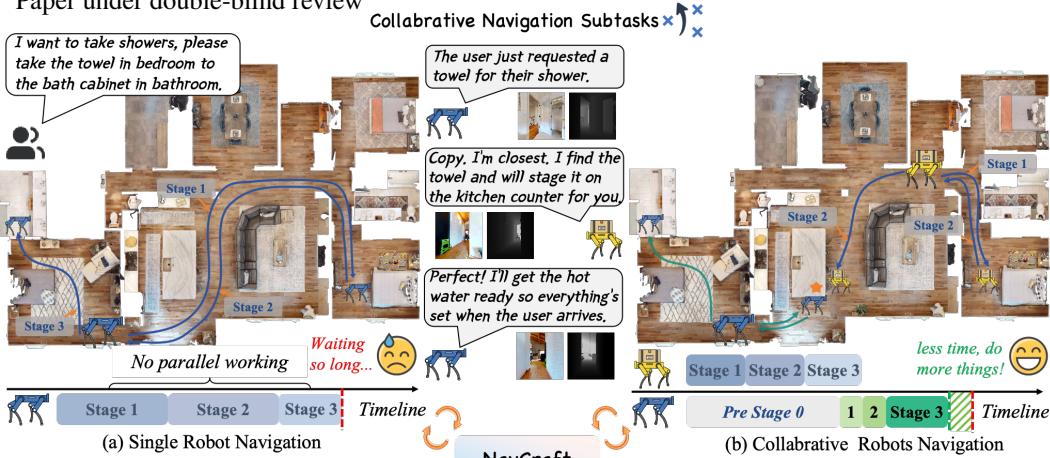


Figure 1: Comparison of single-robot and collaborative multi-robot navigation. (a) In the single-robot case, the agent must complete all stages sequentially, resulting in long delays and idle waiting. (b) In the collaborative case, subtasks are distributed across robots and executed in parallel, reducing overall completion time and enabling the team to accomplish more within the same horizon.

ABSTRACT

Vision-and-Language Navigation (VLN) primarily focuses on a single-agent-centric approach that executes human instructions step-by-step. In real environments with high demand or parallel workflows, collaboration VLN offers distinct benefits including shorter makespan and greater robustness through parallelism and role specialization. Collaboration VLN also brings new challenges including congestion, handoff errors, and rendezvous timing, which single-agent formulations overlook. Current datasets and protocols remain single-agent centered, which hides opportunities for assistance and ignores inter-robot interference. We fill this gap with Collaborative Long-Horizon VLN benchmark (**CoNavBench**), consisting of 4048 single and collaborative episodes with graph-level annotations and a collaboration type taxonomy that controls handoff styles and rendezvous patterns. To generate and evaluate at scale, we build **NavCraft**, an automated graph-grounded data generation platform. A two-stage hierarchical agent first produces a long-horizon base mission for the primary robot and then instantiates helper robots, allocates subgoals, and specifies validated handoffs and rendezvous. The agents operate with a scene graph in the loop derived from Habitat-Sim, which enables reachability checks, travel time, and interference assessment, and iterative schedule repair via an efficiency tool library. As a reference, we provide a collaborative baseline based on a finetuned Qwen2.5-VL-3B. Trained with CoNavBench, collaborative policies reduce makespan and improve reliability over strong single robot counterparts, yielding **18.11%** step level success. Anonymous Website.

1 INTRODUCTION

Vision-and-Language Navigation (VLN) (Anderson et al., 2018; Thomason et al., 2019; Hong et al., 2020) has advanced from stepwise waypoint following to long-horizon, multi-stage settings that demand persistent reasoning and continual re-planning (Khanna et al., 2024). However, most formulations and datasets still assume a single robot agent, suppressing parallelism and ignoring inter-robot

054 Table 1: Comparison to VLN benchmarks. * The scale of our released benchmark is 4048, however
 055 NavCraft is able to generate unlimited data to be tested.

| Benchmark | Simulator | Task Type | Single Task | Collab Task Num | Avg. Gain | Total Task |
|-------------|--------------|-------------------|-------------|-----------------|-----------|------------|
| R2R | Matterport3D | Step-by-step Nav | 21567 | - | - | 21567 |
| REVERIE | Matterport3D | Obj Nav | 21702 | - | - | 21702 |
| VLN-CE | Habitat | Step-by-step Nav | 4475 | - | - | 4475 |
| FAO | Matterport3D | Obj Nav | 3848 | - | - | 3848 |
| Behavior-1k | OmniGibson | Complex Housework | 1000 | - | - | 1000 |
| IVLN | M3D&Habitat | Iterative VLN | 789 | - | - | 789 |
| Goat-Bench | Habitat | Iterative VLN | 725360 | - | - | 725360 |
| LHPR-VLN | Habitat | Multi-stage VLN | 3260 | - | - | 3260 |
| CoNavBench | Habitat | Multi-agent VLN | 2436* | 1612* | 21.08% | 4048* |

066 interference. In contrast, *collaborative* VLN views multiple robots as a coordinated team that ex-
 067 ploits parallelism, anticipates and mitigates inter-robot interference, and optimizes wall-clock time
 068 and energy, which are central to user experience in real deployments (Puig et al., 2023).

069 To close this gap, we introduce Collaborative Long-Horizon VLN benchmark **CoNavBench**, to our
 070 knowledge the first systematic benchmark for collaborative VLN. CoNavBench comprises 4048
 071 single- and multi-robot episodes together with a collaboration type taxonomy controlling handoff
 072 styles and rendezvous patterns. Each episode pairs long-horizon instructions with graph-level anno-
 073 tations, enabling efficient metrics including task success, makespan, and interference time. Given a
 074 long-range instruction, a team must decompose the mission, assign roles, and coordinate handoffs
 075 or rendezvous to minimize time while avoiding congestion.

076 We identify three escalating challenges for collaborative VLN: (i) **Cooperation-ready task synthe-**
 077 **sis**: constructing long-horizon single-robot base tasks with explicit stage boundaries and cross-room
 078 dependencies that create genuine opportunities for assistance (Xu et al., 2022); (ii) **Conflict-free**
 079 **team scheduling**: lifting a base task into a multi-robot schedule with role assignments, temporal
 080 ordering, and rendezvous; and (iii) **In-loop efficiency optimization**: given a feasible schedule, esti-
 081 mating team-level time, anticipating bottlenecks, and issuing actionable guidance.

082 To address the above challenges, we present **NavCraft**, a graph-grounded generation platform
 083 for CoNavBench. NavCraft first constructs a semantically augmented scene graph from Habitat-
 084 Sim (Savva et al., 2019) as the planning blueprint (Rana et al., 2023). Each node is labeled via hier-
 085 archical clustering to assign room categories and functional properties, and edges encode topology
 086 and traversability. Over this spatially grounded representation, a two-stage hierarchical agent oper-
 087 ates: *NavCraft-S* produces a long-horizon single-robot base plan with cross-room scope and explicit
 088 stage structure, and *NavCraft-C* lifts it into a collaborative schedule by instantiating helper robots,
 089 allocating subgoals, and validating handoffs and rendezvous. Unlike text-only prompting (White
 090 et al., 2023), which lacks spatial grounding, and asset-specific simulators (Yang et al., 2024), which
 091 limit throughput and versatility. NavCraft enables context-aware task generation conditioned on
 092 user profiles (Wang et al., 2025b) and robot capabilities, improving diversity and schedule validity.

093 We further propose an **on-graph efficiency tool library** that unifies validation and guidance within
 094 the scene-graph loop. The library translates language intents into numerical constraints over dis-
 095 tances, widths, occlusions, and occupancy; verifies reachability, interference, and estimates time.
 096 It then issues recommendations for subgoal allocation, rendezvous timing, helper deployment, and
 097 route revision. The agent consumes these recommendations in a closed loop, preserving accuracy
 098 while avoiding full-physics rollouts at each step (Wang et al., 2025a).

099 Finally, we provide a reference stack coupling Qwen-series LLMs (Bai et al., 2025) with a memory-
 100 aware mechanism (Song et al., 2025). Policies trained on CoNavBench achieve **18.11%** step level
 101 success than single-robot, indicating a practical path toward deployable collaborative navigation.

103 2 RELATED WORK

104 **Vision-and-Language Navigation** Embodied Vision-and-Language Navigation (VLN) studies
 105 language-conditioned navigation in complex environments. Methods are typically studied in dis-
 106 crete and continuous settings. In discrete VLN (Chen et al., 2022; Zhou et al., 2023), agents move
 107 on a fixed panoramic graph of predefined nodes. The abstraction emphasizes high-level decisions

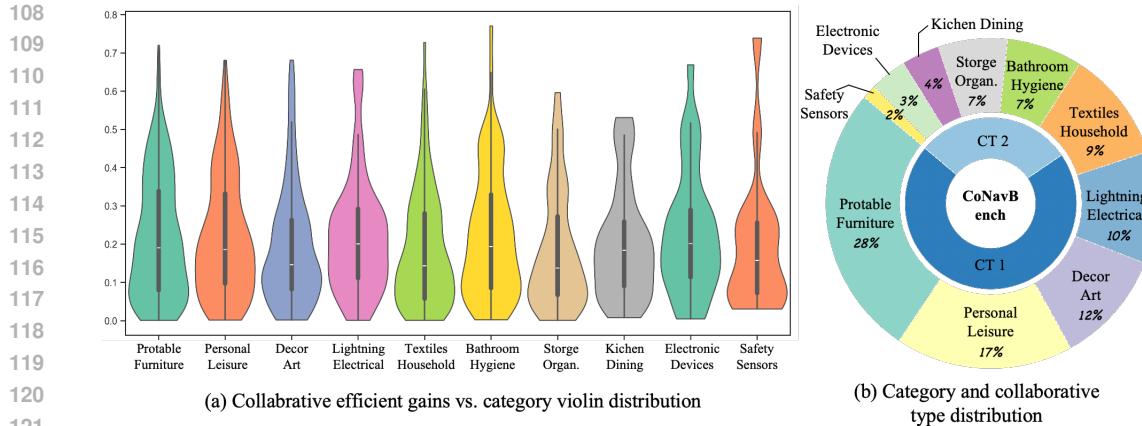
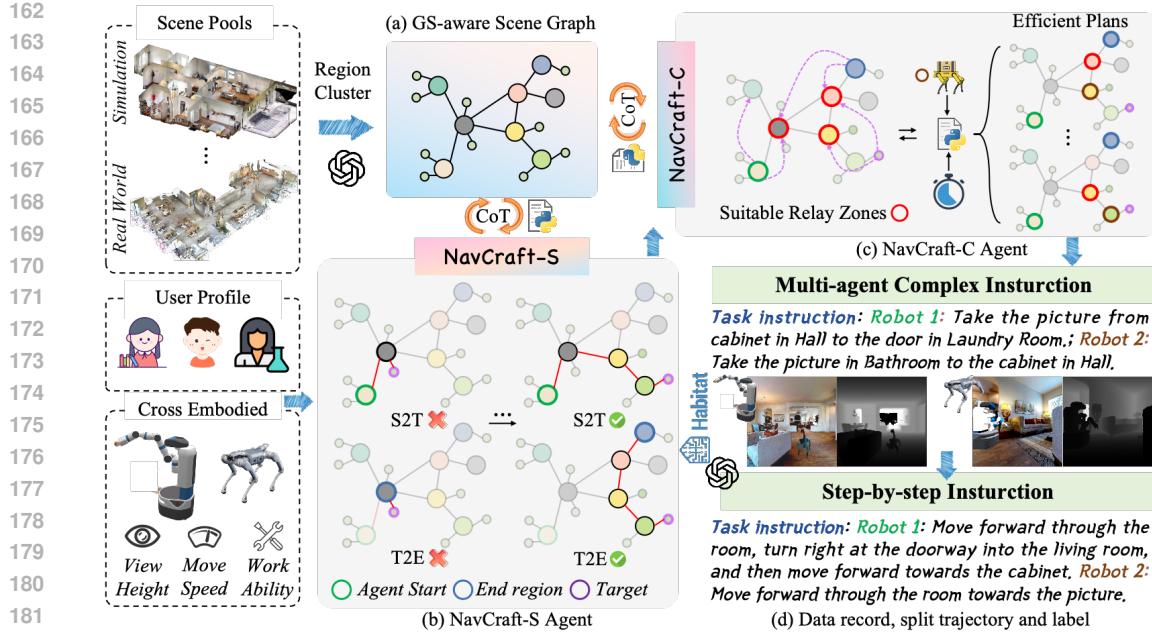


Figure 2: CoNavBench benchmark. **(a) Collaborative efficiency by category.** Violin plots show the distribution of category-wise *efficiency gain* over a single-robot baseline, yielding an average gain of 20% across categories. **(b) Category and collaboration-type distribution.** The benchmark covers a broad and balanced set of household target-object categories (outer ring) and two collaboration types (inner ring), evidencing rich object diversity that supports generalizable evaluation.

while masking metric geometry and collisions. Recent work introduces LLM-augmented planners with retrieval-augmented memory to improve instruction parsing and subgoal proposal (Chen et al., 2024; Zheng et al., 2024), yet these systems still assume oracle connectivity and lack low-level feasibility checks. In continuous environments (Dong et al., 2025), many approaches pretrain waypoint predictors to propose candidate positions for high-level planning (Qiao et al., 2025b; Shi et al., 2025), but such models often overfit and generalize poorly. End-to-end dual-system alternatives reduce this reliance: a high-level planner performs embodied planning with a slow/fast context (Wei et al., 2025), and a low-level controller utilizes a diffusion-policy (Cai et al., 2025) for local motion, improving responsiveness without scene-specific priors. However, across both settings, systems and benchmarks remain predominantly single-agent, with limited modeling of collaboration.

Benchmark for Vision-and-Language Navigation Progress in VLN has been driven by datasets (Qiao et al., 2025a) that steadily raise task complexity and evaluation fidelity. Early datasets, such as R2R (Anderson et al., 2018) and R4R (Jain et al., 2019), study step-by-step instruction following along predefined panoramic trajectories. VLN-CE (Krantz et al., 2020) shifts to continuous control in photorealistic simulators, emphasizing perception and low-level decision making. More recent datasets, including CVDN (Thomason et al., 2019), REVERIE (Qi et al., 2019), and SOON (Zhu et al., 2021), introduce dialogue history, object-centric grounding, and complex instruction comprehension. OVMM (Yenamandra et al., 2023) and Behavior-1K (Li et al., 2022) couple navigation with manipulation and interaction to approximate extended real-world workflows. IVLN (Krantz et al., 2022) and GOAT-Bench (Khanna et al., 2024) enable sequential multi-episode navigation with memory across independent goals, and LHPR-VLN (Song et al., 2025) targets long-horizon planning with multi-stage subtasks in complex indoor environments. Despite this progress, existing benchmarks remain single-agent and lack collaboration primitives (Wang et al., 2023a), which are necessary to study Collaborative VLN with multiple agents subtasks in highly complex environments. This gap motivates our collaborative long-horizon benchmark and platform.

LLM Agents for Vision-and-Language Navigation LLM agents are widely used as policies in interactive domains such as the Web (Chae et al., 2024), games (Hu et al., 2024a), robotics (Zitkovich et al., 2023; Wang et al., 2024b; Cheng et al., 2025), and design (Hu et al., 2024b), where they parse instructions, perceive scenes, and invoke tools. In VLN, these agents typically serve as navigators, grounding observations to produce subgoals or actions. Representative examples include VELMA (Schumann et al., 2023) in Street-View and indoor planners such as NaviLLM (Zheng et al., 2023) via prompts (Saravia, 2022). These systems are single-robot and are evaluated zero-shot or with fine-tuning on existing datasets. However, prior work treats the agent as a navigator (Wang et al., 2024a) rather than a generator. Our departure is to utilize the agent as a data generation and scheduling engine for CoNavBench: NavCraft’s hierarchical agent synthesizes cooperation-ready long-horizon tasks, allocates roles, and produces team-aware schedules.

183 Figure 3: **NavCraft pipeline for CoNavBench benchmark data generation and scheduling.**185

3 PLATFORM, BENCHMARK AND METRICS

187 We introduce NavCraft, a data-generation platform tailored for Collaborative VLN. Using NavCraft,
 188 we construct the CoNavBench benchmark, which enables systematic evaluation of models on long-
 189 horizon, multi-agent planning and execution within vision-language navigation.

191

3.1 NAVCRAFT

193

3.1.1 SCENE GRAPH GENERATION

195 We annotate each node of Habitat connectivity graph $G = (V, E)$ (Wang et al., 2023b) with region
 196 types to obtain a semantic-aware graph. Each node $i \in V$ stores a 3D position $\mathbf{p}_i = [x_i, y_i, z_i]^\top$.
 197 And we also know each region object m 's 3D position $\mathbf{x}_m = [x_m, y_m, z_m]^\top$ and region type $r(m)$.

198 **Instance Proximal Voting** We seed each node by *object-centric* k-NN plurality to preserve coarse-
 199 grained cues. For node i , we search the k nearest annotated objects on the ground plane and assign
 200 the plurality region:

$$201 \quad \mathcal{N}_k(i) = \arg \operatorname{topk}_{m \in \mathcal{M}} \|\mathbf{p}_i - \mathbf{x}_m\|_2, \quad \hat{r}_i^{(0)} = \arg \max_c \sum_{m \in \mathcal{N}_k(i)} \mathbf{1}[r(m) = c].$$

204 where \mathcal{M} is the set of all neighborhood objects, and $k=3$ by default.

205 **Neighborhood Consensus** After seeding node labels with IPV, we observe occasional isolated mis-
 206 labels near narrow passages (e.g., doorways), where object-centric votes can flip a single node at
 207 region boundaries. To address this, we apply a targeted, local correction. A node is eligible for
 208 relabeling only if two guards hold simultaneously: (i) its graph degree is modest, $2 \leq \deg(i) \leq 4$,
 209 and (ii) its provisional label $\hat{r}_i^{(0)}$ disagrees with every neighbor $C(i)$ in G . If both conditions are
 210 met, the node adopts the label of the nearest labeled neighbor on the navigation plane:

$$212 \quad j^* = \arg \min_{j \in C(i)} \|\mathbf{p}_i - \mathbf{p}_j\|_2, \quad \hat{r}_i^{(1)} = \begin{cases} \hat{r}_{j^*}^{(0)}, & \text{if } 2 \leq \deg(i) \leq 4 \text{ and } \forall j \in C(i) : \hat{r}_j^{(0)} \neq \hat{r}_i^{(0)}, \\ \hat{r}_i^{(0)}, & \text{otherwise.} \end{cases}$$

215 **Contiguity Restoration** After IPV and NC seeding, we still observe mislabeled *regions* caused by
 structural barriers such as walls, which fragment a class into multiple small, fractured islands. To

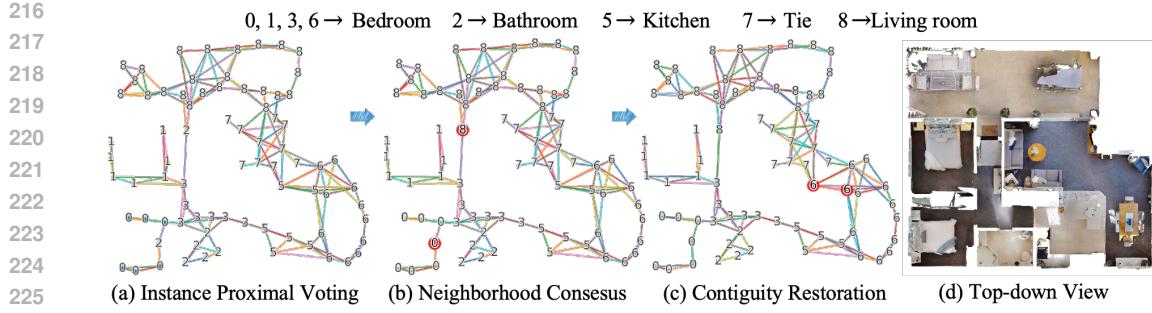


Figure 4: Visualization of the scene graph generation pipeline.

228 restore semantic contiguity without excessive smoothing, we operate at the connected-component
229 level. For each class c , form the induced subgraph $G_c = G[\{i : \hat{r}_i^{(1)} = c\}]$ and keep its largest
230 component and process any remaining component \mathcal{C} as follows. If \mathcal{C} touches no other class, delete
231 it; otherwise, reassign the entire component to the adjacent class with boundary nodes closest to the
232 component centroid:

$$233 \quad \mu_c = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \mathbf{p}_i, \quad c^* = \arg \min_{c'} \frac{1}{|B_{c'}|} \sum_{j \in B_{c'}} \|\mathbf{p}_j - \mu_c\|_2, \quad \hat{r}_i^{(2)} = c^* \quad \forall i \in \mathcal{C}.$$

$$234$$

$$235$$

236 where $B_{c'}$ are boundary neighbors of \mathcal{C} with class c' in G . As shown in Figure 4, the mislabeled
237 '5-Kitchen' island is relabeled into '6-Bedroom' via contiguity restoration.

238 **Graph Contextual Typing** After the preceding steps, a small fraction of regions may still be typed
239 as Unknown. Intuitively, these are ambiguous areas where local object votes and connectivity cues
240 are not confident enough on their own. For any region type still Unknown, we make a single pass
241 that combines graph context and object inventory. We build a compact summary: (i) the histogram
242 of adjacent room types in G , and (ii) the top-5 object names in that region and query a lightweight
243 instruction-following model h_ϕ for the most plausible type; otherwise, we keep the current label:

$$244 \quad \hat{r}_i^{\text{final}} = \begin{cases} h_\phi(\text{adjacent-type hist, top-5 objects}), & \text{if } \hat{r}_i^{(2)} = \text{Unknown}, \\ \hat{r}_i^{(2)}, & \text{otherwise.} \end{cases}$$

$$245$$

$$246$$

247 where h_ϕ is the GPT4o-mini and its output updates the per-scene region dictionary, while node
248 positions and edges remain unchanged.

250 3.1.2 NAVCRAFT-S

251 **Goal** Given the room-labeled connectivity graph and per-scene item lists, NavCraft-S selects a fea-
252 sible triple: start region s , target-object region t , end region e and validates it with graph constraints,
253 and synthesizes a directed region-to-region path. A *user profile* π is injected via the prompt as a
254 *light preference* when sampling target objects, and it never overrides feasibility.

255 **Profile-conditioned sampling** We simulate user demand with a lightweight role profile π that en-
256 codes age, occupation, and lifestyle. The profile is injected into the prompt to encourage diverse
257 habits and phrasing, and it is used only as a tie-breaker when multiple objects or destinations are
258 equally eligible. Following the role templates in NavRAG (Wang et al., 2025b), NAVCRAFT-S
259 then samples a portable object and compatible start and end regions conditioned on π . This simple
260 conditioning increases variety without relying on hand-crafted priors.

261 **Feasibility** Given a candidate triple (s, t, e) , NavCraft-S must first ensure that the underlying naviga-
262 tion problem is meaningful: each leg should be reachable on the region graph and long enough to
263 span multiple rooms. As shown in Figure 3, we evaluate the two legs $s \rightarrow t$ and $t \rightarrow e$ on the region
264 graph using the skill library. Let $L(u, v) = \text{dist}_H(u, v)$ denote hop distance on the region graph,
265 $\text{conn}(u, v)$ indicate reachability, and $\text{adj}(u, v) \iff L(u, v) = 1$. We introduce a hop threshold
266 $\tau \geq 1$ to control the minimum cross-room extent (CoNavBench sets $\tau = 2$). A leg is admissible if it
267 is connected and non-trivial in length. To avoid redundancy with the non-adjacency rule, we enforce
268

$$269 \quad \text{leg_ok}(u, v) := \text{conn}(u, v) \wedge L(u, v) \geq \max\{2, \tau\},$$

$$\text{valid} := \text{leg_ok}(s, t) \wedge \text{leg_ok}(t, e).$$

270 In words, each leg must be reachable and span at least $\max\{2, \tau\}$ hops, and larger τ encourages
 271 longer-range plans. Once valid holds, we concatenate hop-shortest paths for the two legs and com-
 272 press consecutive nodes that belong to the same region to obtain region transitions:
 273

$$\text{Path}(s \rightsquigarrow e) = \text{SP}_H(s, t) \oplus \text{SP}_H(t, e).$$

275 3.1.3 NAVCRAFT-C

276 Given the high-level triple (s, t, e) and the region path produced by NAVCRAFT-S, NAVCRAFT-C
 277 lifts the single agent plan to a collaborative execution on the same semantic and geometric graph.
 278 The module first decides whether collaboration is beneficial. If collaboration is selected, it fixes a
 279 collaboration type and hands off. Details of skill templates appear in the Appendix.
 280

281 **Type abstraction** We use two canonical handoff patterns that allow the planner to reason about
 282 cooperation independently of motion primitives. *Type A1*: the collaborator picks up the object in
 283 region t , hands off at transfer region x , and the main agent delivers from x to e . *Type A2*: the main
 284 agent picks up in t , hands off at x , and the collaborator delivers from x to e .

285 **Augmented metric graph** Let \mathcal{T} be the set of region nodes and $\mathcal{A} = \{a_e, a_x\}$ the set of anchors
 286 (end asset, candidate transfer asset). For any anchor $a \in \mathcal{A}$, let $c(a) \in \mathcal{T}$ denote the region that
 287 contains a . We build an augmented node graph G^+ from the Habitat connectivity graph G . Edges
 288 in G^+ use 2D Euclidean weights. Each anchor a is inserted at its physical location and linked to the
 289 nearest navigable node. We then use a single distance on G^+ :

$$d(x, y) = \text{dist}_{G^+}(x, y).$$

290 **Collaborative Generation** We quantify when it is worthwhile to involve a second agent by comparing
 291 how much travel load the main agent would bear alone versus under different collaboration
 292 patterns. Intuitively, a collaboration is only accepted if bringing in the helper strictly shortens the
 293 main agent’s own route. The single-agent baseline load borne by the main agent is:
 294

$$C_{\text{solo}} = d(s, o) + d(o, a_e).$$

295 Given a candidate tuple $(type, a_x)$, we evaluate the main agent load under two types of collaboration:
 296

$$J_{r_1}^{\text{A1}} = d(s, a_x) + d(a_x, a_e), \quad J_{r_1}^{\text{A2}} = d(s, t) + d(t, a_x).$$

297 Here J_{r_1} denotes the load borne by the main agent, and the A1/A2 indexes the collaboration types.
 298 We accept collaboration if:

$$\min\{J_{r_1}^{\text{A1}}, J_{r_1}^{\text{A2}}\} < C_{\text{solo}},$$

299 and we report the improvement ratio $\alpha = \min\{J_{r_1}^{\text{A1}}, J_{r_1}^{\text{A2}}\}/C_{\text{solo}}$. A candidate must satisfy the
 300 scene-graph guards: (i) $x \neq t$ and a_x exists under x ; (ii) both agents can reach x in G^+ ; (iii) the
 301 collaborator’s start follows non-adjacency and connectedness rules to (t, x) . The planner iteratively
 302 proposes tuples and records them with $\alpha < 1$.
 303

304 3.2 THE CONAVBENCH BENCHMARK AND METRICS

305 **Benchmark Definition** The proposed CoNavBench benchmark is designed to evaluate collabora-
 306 tive embodied navigation under multi-agent settings. Unlike conventional single-agent VLN tasks,
 307 where an agent must complete a long-horizon instruction end-to-end, CoNavBench decomposes a
 308 complex instruction into multiple collaborative subtasks (referred to as *collab-stages*). A typical
 309 high level user command follows the pattern: “*Find object A at location X, and deliver it to location Y*”. Instead of requiring a single agent to traverse and manipulate across the entire trajectory, the
 310 task is distributed across two agents. Concretely, one agent is responsible for locating and transport-
 311 ing the target to an intermediate relay point, after which another agent continues the delivery to the
 312 final goal. As shown in Figure 5, this decomposition produces coordinated trajectories with a clear
 313 relay handoff between agents. Compared to the single-agent rollout in Figure 5.(a), the collabora-
 314 tive setting in Figure 5.(b) reduces backtracking and shortens paths by assigning complementary explo-
 315 ration regions. First-person views in Figure 5.(c) further illustrate the handoff and mutual avoidance
 316 behaviors that enable efficient, reliable delivery. This decomposition mirrors realistic multi-robot co-
 317 operation, mitigates memory overflow issues in long-horizon reasoning, and empirically improves
 318 task success rates. Meanwhile, we also follow (Song et al., 2025) to decompose the high-level tasks
 319 and create step-by-step VLN tasks for each trajectory segment to alleviate the inherent difficulty of
 320 executing the abstract instructions.
 321

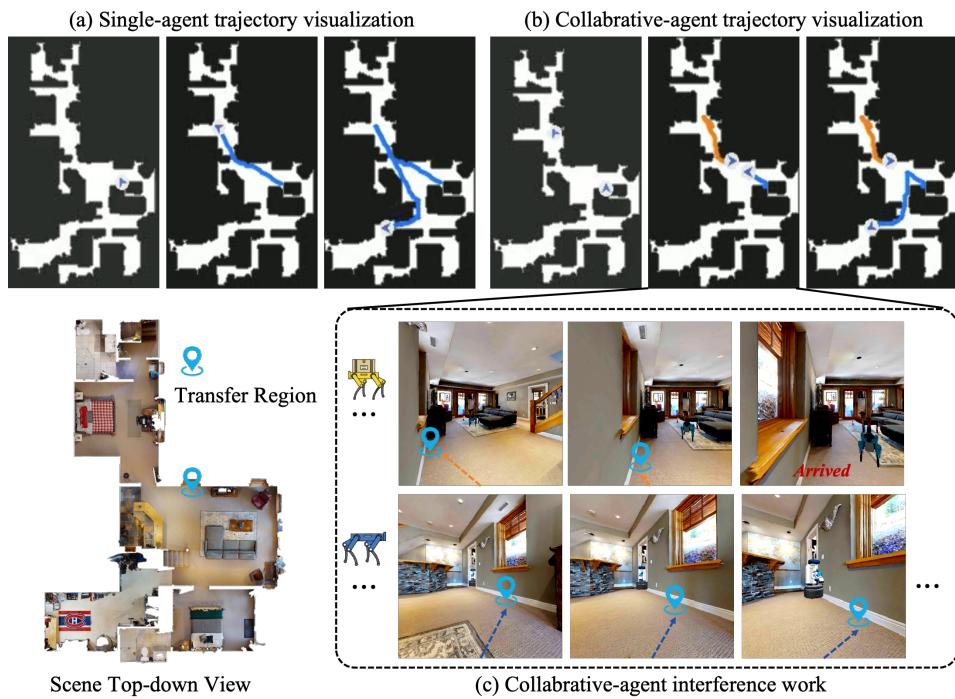


Figure 5: (a) Single-agent trajectory visualization: the agent navigates independently following the given instruction. (b) Collaborative-agent trajectory visualization: two agents cooperate during navigation, showing more efficient and coordinated exploration paths. (c) Collaborative-agent interference work: visual examples from the agents’ first-person perspectives, illustrating interaction and coordination in shared environments.

Evaluation Metrics To rigorously measure performance on CoNavBench, we employ standard navigation metrics: (i) **Success Rate (SR)**, the percentage of episodes where the agent(s) successfully complete the task within a 1.0 m goal threshold; (ii) **Success weighted by Path Length (SPL)**, which normalizes success by the efficiency of the trajectory; and (iii) **Navigation Error**, computed as the geodesic distance between the agent’s final position and the goal when the task terminates. In addition, we extend evaluation with two subtask metrics originally proposed in LH-VLN (Song et al., 2025): **Independent Completion Rate (ICR)** quantifies the success of each sub-task individually, thereby providing insight into the robustness of agents when executing isolated segments of the collaborative pipeline. **Conditional Success Rate (CSR)** measures the overall success of the full multi-agent task, where completion depends on all preceding subtasks being successfully executed. CSR thus captures interdependencies across collab-stages and reflects the degree to which agents can coordinate seamlessly over extended task horizons.

4 EXPERIMENT

4.1 EXPERIMENTAL SETTING

Simulator Settings All experiments are conducted in HABITAT3 (Puig et al., 2023), a continuous 3D simulation platform for vision-and-language navigation (VLN). Unless otherwise specified, each agent is equipped with synchronized RGB and depth sensors mounted in three directions: front, left (+60°), and right (-60°). To ensure comparability with prior work, we adopt the atomic action space used in LH-VLN (Song et al., 2025): *move forward* (+0.25 m), *turn left* (+30°), *turn right* (-30°), and *stop*. An episode (or sub-episode) is regarded as completed either when the agent issues *stop* or when the geodesic distance to the designated target falls below 1.0 m.

Embodied We instantiate two articulated robot embodiments from URDF models: the Fetch mobile manipulator and Boston Dynamics Spot. Fetch features a wheeled base with an upper-body manipulator and structural frame; Spot is a quadruped robot capable of carrying a back-mounted

378
379
380
381 Table 2: Performance comparison on the **Single-Agent Task** in CoNavBench. Results are shown
382 for both high-level tasks and step-by-step subtasks.
383
384
385
386
387

| Method | Type | Single Agent Task (High-level/Step-by-step Tasks) | | | | |
|----------------|-----------|---|------------|-------------|-------------|-----------|
| | | SR↑ | SPL↑ | ISR↑ | CSR↑ | NE↓ |
| Random | - | 0.00/1.26 | 0.00/1.26 | 1.61/1.26 | 1.21/1.26 | 7.25/7.56 |
| Qwen2.5-VL-3B* | Zero-shot | 4.30/10.41 | 0.98/2.55 | 14.25/10.41 | 16.10/10.41 | 6.91/7.80 |
| | Finetuned | 12.90/29.65 | 6.08/13.81 | 23.92/29.65 | 26.22/29.65 | 6.40/6.74 |
| Qwen2.5-VL-7B* | Zero-shot | 0.00/1.26 | 0.00/1.26 | 1.84/1.26 | 1.33/1.26 | 7.21/7.56 |
| | Finetuned | 10.22/22.40 | 4.93/12.57 | 22.58/22.40 | 22.45/22.40 | 6.39/7.46 |

388
389
390
391 Table 3: Performance comparison on the **Collaborative-Agent Task** in CoNavBench. Results are
392 shown for both high-level tasks and step-by-step subtasks.
393
394

| Method | Type | Collaborative Agent Task (High-level/Step-by-step Tasks) | | | | |
|----------------|-----------|--|------------|-------------|-------------|-----------|
| | | SR↑ | SPL↑ | ISR↑ | CSR↑ | NE↓ |
| Random | - | 0.00/3.43 | 0.00/3.43 | 2.30/3.43 | 1.76/3.43 | 7.15/6.35 |
| Qwen2.5-VL-3B* | Zero-shot | 8.67/19.86 | 2.89/5.44 | 16.12/19.86 | 16.12/19.86 | 6.82/6.50 |
| | Finetuned | 11.11/35.02 | 4.82/16.88 | 20.19/35.02 | 20.12/35.02 | 6.55/5.79 |
| Qwen2.5-VL-7B* | Zero-shot | 0.00/3.61 | 0.00/3.61 | 1.90/3.61 | 1.36/3.61 | 7.12/6.35 |
| | Finetuned | 11.65/29.78 | 6.24/16.56 | 21.00/29.78 | 20.93/29.78 | 6.74/6.20 |

400
401 arm. These embodiments allow us to test navigation and embodied interaction under heterogeneous
402 morphology and locomotion dynamics, while keeping sensing and policy stacks consistent.403
404 **Scene Assets** Our environments are primarily drawn from HM3D (Ramakrishnan et al., 2021), com-
405 prising 216 large-scale, semantically annotated indoor reconstructions. We adopt the scene graph
406 initialization from SCALEVLN to provide object- and room-level structure. In addition, we develop
407 a *real2sim* pipeline that scans real-world indoor spaces and imports them into Habitat, enabling
408 closed-loop validation of our data generation with NAVCRAFT. More details are in the Appendix.409
410 **Baselines and Training Settings** We follow a trajectory-supervised learning paradigm. For mul-
411 timodal reasoning, we employ the Qwen-2.5VL family and report results for both small- and mid-
412 scale variants (3B/7B), evaluated in zero-shot settings and after fine-tuning on the CoNavBench
413 corpus. Visual features are extracted by a ViT backbone from EVA-CLIP-02-LARGE; the visual
414 encoder remains *frozen* during all training runs to stabilize optimization and reduce compute. Un-
415 less noted otherwise, we fine-tune the language-conditioned policy and control heads (full-parameter
416 fine-tuning on the non-visual modules), using Adam with a learning rate of 3×10^{-5} . Training is
417 performed on four NVIDIA A800 GPUs with a per-step batch size of 1; a complete run typically
418 finishes in about four days. We apply standard practices for reproducibility: fixed random seeds,
419 gradient clipping, and validation-based early stopping. More details are in the Appendix.420
421

4.2 RESULT AND ANALYSIS

422
423 **Single Agent Performance** Table 2 reports single-agent results under both the high-level tasks and
424 step-by-step protocols. Random policies fail across all metrics, confirming the benchmark’s non-
425 trivial difficulty. Zero-shot Qwen2.5-VL models perform slightly above random on SR and ISR yet
426 remain far from practical utility, with task-level SR below 5%, indicating that high-level instruc-
427 tions are hard to follow. After finetuning, both Qwen2.5-VL-3B and 7B show clear absolute gains:
428 SR improves at both the high-level and step-by-step settings, accompanied by consistent increases
429 in SPL, ISR, and CSR, and a reduction in navigation error. Despite these improvements, abso-
430 lute performance remains modest, underscoring that CoNavBench poses a challenging single-agent
431 benchmark and leaving ample headroom for future methods.432
433 **Collaborative Agent Performance** Table 3 compares collaboration with single-agent execution.
434 Under the **step-by-step** protocol, both Qwen2.5-VL-3B and 7B improve: the finetuned 3B model
435 raises SR from 29.65% to 35.02%, and the finetuned 7B model rises from 22.40% to 29.78% with
436 SPL increasing from 12.57 to 16.56 and NE decreasing, consistent with CoNavBench’s design that

432
433 Table 4: Performance, efficiency, cost and latency of NAVCRAFT-powered agents. Higher numbers
434 are better (\uparrow) except Cost (\downarrow). Note that the success rate represents task generation.

| 435 436 437 Powered Agent | 438 Success Rate \uparrow | | 439 Collab Gain \uparrow | | 440 Cost (\$) \downarrow | 441 Sample Eff. (s/iter) \downarrow |
|------------------------------------|--------------------------------|---------------|-------------------------------|------------|-------------------------------|---|
| | 442 Single | 443 Collab | 444 Max | 445 Avg | | |
| <i>Google</i> | | | | | | |
| 2.0-flash | 47% | 8.51% | 29.72% | 22.32% | 0.265 | 16.82 |
| 2.5-flash-nothink | 41% | 9.76% | 13.18% | 9.24% | 0.902 | 18.71 |
| <i>Claude</i> | | | | | | |
| 3-5-haiku | 51% | 3.90% | 50.07% | 37.06% | 1.142 | 14.94 |
| <i>OpenAI</i> | | | | | | |
| 4o | 77% | 46.75% | 75.23% | 21.07% | 5.242 | 30.85 |
| 4o-mini | 64% | 26.56% | 47.32% | 25.12% | 0.360 | 21.41 |
| 4.1-mini | 68% | 23.53% | 42.80% | 19.10% | 0.494 | 17.92 |

446 splits a long multi-stage instruction into single-stage subtasks and shortens the decision horizon.
447 In the **high-level** end-to-end evaluation, gains are smaller and some metrics slightly regress, partly
448 because test-time relay planning introduces intermediate handoff points and auxiliary phrasing that
449 can cause mild vision-language mismatches, and partly because high-level completion is inherently
450 difficult: success compounds across stages (CSR), early errors propagate, and coordination overhead
451 from state synchronization and re-localization enlarges the effective search space. Finally, although
452 the 7B model is competitive, it does not surpass 3B under the current data budget, which suggests
453 under-training rather than a fundamental limitation; viewed in isolation, the collaborative setting still
454 benefits 7B (SR **22.40%** to **29.78%**, SPL **12.57** to **16.56**, NE decreases), indicating that multi-agent
455 decomposition reliably improves local competence even when model capacity is not fully exploited.
456

457 4.3 ABLATION STUDIES

458 Table 4 evaluates representative off-the-shelf agents from Google, Claude, and OpenAI under both
459 single and collaborative settings, revealing clear capability and cost-efficiency trade-offs. Collaboration
460 generally improves performance but with varying magnitude across families. Claude-3.5-
461 haiku shows the second relative collaboration gain 50.07% despite very low absolute collaborative
462 task generation success rate 3.90%, suggesting that weaker agents can benefit from task decomposi-
463 tion but still fail to achieve reliable success. OpenAI’s 4o achieves the strongest absolute results with
464 77% in the single-agent and 46.75% success rate in the collaborative task generation, while Google’s
465 models provide lower cost per sample but limited task generation success rate, which indicates that
466 efficiency alone cannot compensate for weak grounding ability. Among these choices, GPT-4o-
467 mini offers the most favorable balance: it reaches 26.56% collaborative task generation SR and the
468 highest average collaboration gains among the OpenAI variants 25.12% at substantially lower cost,
469 about 0.360 per sample compared to 5.242 for 4o. Based on this analysis, we adopt GPT-4o-mini as
470 the data generation agent for CoNavBench, allowing us to scale instruction and trajectory synthesis
471 while maintaining a strong balance between quality and price efficiency.

472 **Limitation** Our framework still has several limitations. First, the data generation pipeline relies on
473 GPT API models, which can introduce stylistic bias and make reproducibility sensitive to backend
474 updates; a natural next step is to train or distill an open LLM specialized for CoNavBench and to
475 improve throughput with batching, caching, and on-graph pruning. Second, NavCraft-C currently
476 targets two-agent relay patterns in indoor HM3D-style scenes, so coverage of richer collaboration
477 and three or more robots remains limited, partly due to scene size.

478 5 CONCLUSION

479 We introduced **CoNavBench**, a collaborative vision-and-language navigation benchmark with 4048
480 episodes, a collaboration type taxonomy, and graph-level annotations that enable team-aware eval-
481 uation (success, makespan, energy, and interference time). To populate and study this setting, we pre-
482 sented **NavCraft**, a graph-grounded generation platform built on a semantically augmented scene-

graph substrate, along with a two-stage agent (*NavCraft-S* and *NavCraft-C*) and an on-graph efficiency tool library for closed-loop validation and guidance. Beyond establishing the benchmark, we provide a reference stack that couples Qwen-series LLMs. Policies trained on CoNavBench achieve **18.11%** step-level task success rate compared to single-robot, indicating a practical path toward deployable collaborative vision-and-language navigation.

492 REFERENCES

494 Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid,
 495 Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting
 496 visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3674–3683, 2018.

498 Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang,
 499 Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*,
 500 2025.

502 Wenzhe Cai, Jiaqi Peng, Yuqiang Yang, Yujian Zhang, Meng Wei, Hanqing Wang, Yilun Chen,
 503 Tai Wang, and Jiangmiao Pang. Navdp: Learning sim-to-real navigation diffusion policy with
 504 privileged information guidance. *arXiv preprint arXiv:2505.08712*, 2025.

505 Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim,
 506 Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and
 507 leveraging environment dynamics in web navigation. *arXiv preprint arXiv:2410.13232*, 2024.

509 Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee K. Wong. Mapgpt:
 510 Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

512 Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think
 513 global, act local: Dual-scale graph transformer for vision-and-language navigation. 2022
 514 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16516–16526,
 515 2022.

517 An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem
 518 Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action
 519 model for navigation. In *RSS*, 2025.

520 Yifei Dong, Fengyi Wu, Qi He, Heng Li, Minghan Li, Zebang Cheng, Yuxuan Zhou, Jingdong
 521 Sun, Qi Dai, Zhi-Qi Cheng, et al. Ha-vln: A benchmark for human-aware navigation in discrete-
 522 continuous environments with dynamic multi-human interactions, real-world validation, and an
 523 open leaderboard. *arXiv preprint arXiv:2503.14229*, 2025.

525 Michael Hahsler, Matt Piekenbrock, and Derek Doran. dbSCAN: Fast density-based clustering with
 526 r. *Journal of Statistical Software*, 2019.

527 Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln-bert: A
 528 recurrent vision-and-language bert for navigation. 2021 *IEEE/CVF Conference on Computer
 529 Vision and Pattern Recognition (CVPR)*, pp. 1643–1653, 2020.

531 Sihao Hu, Tiansheng Huang, Gaowen Liu, Ramana Rao Kompella, Fatih Ilhan, Selim Furkan Tekin,
 532 Yichang Xu, Zachary Yahn, and Ling Liu. A survey on large language model-based game agents.
 533 *arXiv preprint arXiv:2404.02039*, 2024a.

534 Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and
 535 Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *Forty-first
 536 International Conference on Machine Learning*, 2024b.

538 Vihan Jain, Gabriel Ilharco, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay
 539 on the path: Instruction fidelity in vision-and-language navigation. In *Annual Meeting of the
 Association for Computational Linguistics*, 2019.

540 Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Théophile Gervet,
 541 Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi.
 542 Goat-bench: A benchmark for multi-modal lifelong navigation. *2024 IEEE/CVF Conference
 543 on Computer Vision and Pattern Recognition (CVPR)*, pp. 16373–16383, 2024.

544

545 Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-
 546 graph: Vision-and-language navigation in continuous environments. In *European Conference on
 547 Computer Vision*, pp. 104–120. Springer, 2020.

548

549 Jacob Krantz, Shurjo Banerjee, Wang Zhu, Jason J. Corso, Peter Anderson, Stefan Lee, and Jesse
 550 Thomason. Iterative vision-and-language navigation. *2023 IEEE/CVF Conference on Computer
 551 Vision and Pattern Recognition (CVPR)*, pp. 14921–14930, 2022.

552

553 Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-
 554 Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune
 555 Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan
 556 Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio
 557 Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A benchmark
 558 for embodied ai with 1, 000 everyday activities and realistic simulation. In *Conference on Robot
 Learning*, 2022.

559

560 Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang,
 561 Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, Vladimir Von-
 562 drus, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakri-
 563 shnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and
 564 Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

565

566 Yuankai Qi, Qi Wu, Peter Anderson, Xin Eric Wang, William Yang Wang, Chunhua Shen, and
 567 Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor
 568 environments. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,
 pp. 9979–9988, 2019.

569

570 Yanyuan Qiao, Haodong Hong, Wenqi Lyu, Dong An, Siqi Zhang, Yutong Xie, Xinyu Wang, and
 571 Qi Wu. Navbench: Probing multimodal large language models for embodied navigation. *ArXiv*,
 572 abs/2506.01031, 2025a.

573

574 Yanyuan Qiao, Wenqi Lyu, Hui Wang, Zixu Wang, Zerui Li, Yuan Zhang, Mingkui Tan, and Qi Wu.
 575 Open-nav: Exploring zero-shot vision-and-language navigation in continuous environment with
 576 open-source llms. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*,
 pp. 6710–6717. IEEE, 2025b.

577

578 Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander
 579 Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang,
 580 Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-
 581 scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Pro-
 582 cessing Systems Datasets and Benchmarks Track*, 2021. URL <https://arxiv.org/abs/2109.08238>.

583

584 Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian D. Reid, and Niko Sünderhauf.
 585 Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. In
 586 *Conference on Robot Learning*, 2023.

587

588 Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham
 589 Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Va-
 590 sudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Fe-
 591 ichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*,
 592 2024. URL <https://arxiv.org/abs/2408.00714>.

593

Elvis Saravia. Prompt Engineering Guide. <https://github.com/dair-ai/Prompt-Engineering-Guide>,
 12 2022.

594 Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain,
 595 Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied
 596 ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp.
 597 9339–9347, 2019.

598

599 Raphael Schumann, Wanrong Zhu, Weixi Feng, Tsu-Jui Fu, Stefan Riezler, and William Yang Wang.
 600 Velma: Verbalization embodiment of llm agents for vision and language navigation in street view.
 601 In *AAAI Conference on Artificial Intelligence*, 2023.

602

603 Xiangyu Shi, Zerui Li, Wenqi Lyu, Jiatong Xia, Feras Dayoub, Yanyuan Qiao, and Qi Wu. Smart-
 604 way: Enhanced waypoint prediction and backtracking for zero-shot vision-and-language naviga-
 605 tion. *ArXiv*, abs/2503.10069, 2025.

606

607 Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-
 608 horizon vision-language navigation: Platform, benchmark and method. In *Proceedings of the*
 609 *Computer Vision and Pattern Recognition Conference*, pp. 12078–12088, 2025.

610

611 Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navi-
 612 gation. In *Conference on Robot Learning*, 2019.

613

614 Jiawei Wang, Renhe Jiang, Chuang Yang, Zengqing Wu, Makoto Onizuka, Ryosuke Shibasaki, and
 615 Chuan Xiao. Large language models as urban residents: An llm agent framework for personal
 616 mobility generation. *ArXiv*, abs/2402.14744, 2024a.

617

618 Liuyi Wang, Xinyuan Xia, Hui Zhao, Hanqing Wang, Tai Wang, Yilun Chen, Chengju Liu, Qijun
 619 Chen, and Jiangmiao Pang. Rethinking the embodied gap in vision-and-language navigation: A
 620 holistic study of physical and visual disparities. *arXiv preprint arXiv:2507.13019*, 2025a.

621

622 Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li,
 623 and José M. Álvarez. Omnidrive: A holistic llm-agent framework for autonomous driving with
 624 3d perception, reasoning and planning. *ArXiv*, abs/2405.01533, 2024b.

625

626 Tianhang Wang, Guang Chen, Kai Chen, Zhengfa Liu, Bo Zhang, Alois Knoll, and Changjun Jiang.
 627 Umc: A unified bandwidth-efficient and multi-resolution based collaborative perception frame-
 628 work. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8187–
 629 8196, 2023a.

630

631 Zihan Wang, Yaohui Zhu, Gim Hee Lee, and Yachun Fan. Navrag: Generating user demand instruc-
 632 tions for embodied navigation through retrieval-augmented llm. *arXiv preprint arXiv:2502.11142*,
 633 2025b.

634

635 Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and
 636 Yu Qiao. Scaling data generation in vision-and-language navigation. In *ICCV 2023*, 2023b.

637

638 Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu,
 639 Wenzhe Cai, Hanqing Wang, Yilun Chen, et al. Streamvln: Streaming vision-and-language navi-
 640 gation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

641

642 Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf El-
 643 nashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance
 644 prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

645

646 Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. Opv2v: An open benchmark
 647 dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *2022 IEEE*
 648 *International Conference on Robotics and Automation (ICRA)*, 2022.

649

650 Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu,
 651 Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d
 652 embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and*
 653 *Pattern Recognition*, pp. 16227–16237, 2024.

648 Sriram Yenamandra, A. Ramachandran, Karmesh Yadav, Austin S. Wang, Mukul Khanna,
649 Théophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander Clegg, John Turner, Zsolt Kira, Mano-
650 lis Savva, Angel X. Chang, Devendra Singh Chaplot, Dhruv Batra, Roozbeh Mottaghi, Yonatan
651 Bisk, and Chris Paxton. Homerobot: Open-vocabulary mobile manipulation. In *Conference on*
652 *Robot Learning*, 2023.

653 Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. Towards learning a generalist
654 model for embodied navigation. *2024 IEEE/CVF Conference on Computer Vision and Pattern*
655 *Recognition (CVPR)*, pp. 13624–13634, 2023.

656 Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. Towards learning a generalist
657 model for embodied navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision*
658 *and Pattern Recognition*, pp. 13624–13634, 2024.

659 Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navi-
660 gation with large language models. In *AAAI Conference on Artificial Intelligence*, 2023.

661 Fengda Zhu, Xiwen Liang, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Soon: Scenario oriented
662 object navigation with graph-based exploration. *2021 IEEE/CVF Conference on Computer Vision*
663 *and Pattern Recognition (CVPR)*, pp. 12684–12694, 2021.

664 Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart,
665 Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge
666 to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702 **A APPENDIX**
703704 **APPENDIX LISTS**
705

- 706 • Real2Sim ToolBox
- 707 • Experimental Setup
- 708 • Supplementary Experiments
- 709 • Discussion
- 710 • Large Language Models Usage Statement
- 711 • Ethics statement
- 712 • Human vs. NavCraft Case Study
- 713 • User Profile
- 714 • Real Robot Demo
- 715 • Implementation details of NavCraft-S
- 716 • Implementation details of NavCraft-C
- 717 • Prompts for Graph contextual Typing
- 718 • Ablation for Prompts

723 **A.1 REAL2SIM TOOLBOX**
724

734 (a) Scene Mesh via Scanner App

734 (b) 3D Bounding boxes of target objects

734 (c) Navmesh

735 Figure 6: Pipeline of Real-world into NavCraft data generation.
736737 **Algorithm 1** Real2Sim ToolBox

738 **Require:** $M_{\text{iOS}}, \{I_t, K_t, R_t, \mathbf{t}_t\}_{t=1}^T$
 739 **Ensure:** Object list \mathcal{O} , navigable graph \mathcal{N} , scene graph \mathcal{G} , tasks from NavCraft-S/C

740 1: $M_{\text{align}} \leftarrow T_{\text{align}} \cdot M_{\text{iOS}}; [R_t | \mathbf{t}_t] \leftarrow T_{\text{align}} \cdot [R_t | \mathbf{t}_t], \forall t$
 741 2: **for** $t = 1$ to T **do**
 742 3: $\mathcal{B}_t \leftarrow \text{SegmentAnything}(I_t)$ ▷ 2D instance masks & boxes
 743 4: **for** each $b \in \mathcal{B}_t$ **do**
 744 5: $\mathcal{P}_{t,b} \leftarrow \text{BackProjectToMesh}(b, K_t, R_t, \mathbf{t}_t, M_{\text{align}})$
 745 6: $\hat{o}_{t,b} \leftarrow \text{Fit3DBox}(\mathcal{P}_{t,b})$ ▷ provisional 3D proposal
 746 7: **end for**
 747 8: **end for**
 748 9: $\mathcal{O} \leftarrow \text{DBSCAN_Merge}(\{\hat{o}_{t,b}\}_{t,b})$ ▷ merge across views using 3D position/size + 2D IoU cues
 749 10: $\mathcal{N} \leftarrow \text{GenerateNavmeshAndNodes}(M_{\text{align}})$ ▷ via scaleVNL scripts
 750 11: $\mathcal{G} \leftarrow \text{BuildSceneGraph}(\mathcal{N}, \mathcal{O})$ ▷ rooms/objects/portals; topology & traversability attributes
 751 12: $\text{Plan}_S \leftarrow \text{NavCraft-S}(\mathcal{G})$ ▷ single-robot base plan with stages
 752 13: $\text{Plan}_C \leftarrow \text{NavCraft-C}(\mathcal{G}, \text{Plan}_S)$ ▷ collaborative schedule with validated handoffs/rendezvous
 753 14: **return** $\mathcal{O}, \mathcal{N}, \mathcal{G}, \text{Plan}_S, \text{Plan}_C$

754
 755 **Goal** Lift a real indoor scene into a NavCraft-ready, semantics-geometry-aware scene graph that
 supports task generation and collaborative scheduling.

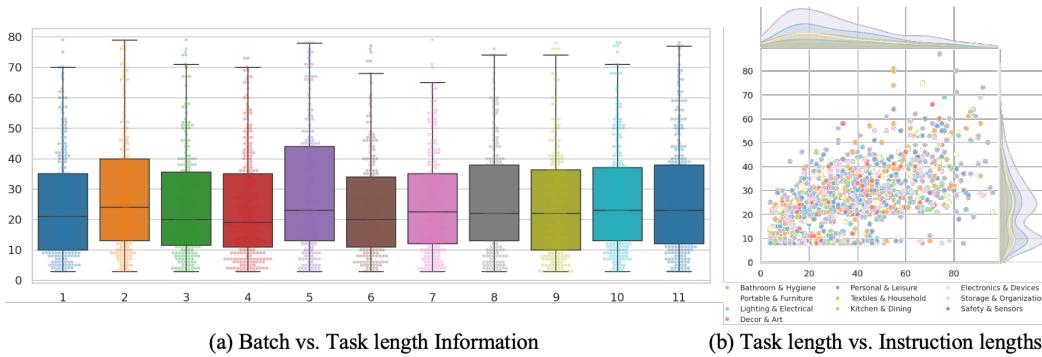


Figure 7: (a) Batch vs. Task length information across different data splits. Each boxplot illustrates the distribution of task lengths within a specific batch (1–11). (b) Task length vs. instruction lengths across different domains, showing their joint distribution with density overlays.

Inputs (i) An iOS LiDAR scan (triangle mesh M_{iOS}) with keyframe RGB images $\{I_t\}$, camera intrinsics $\{K_t\}$, and extrinsics (world-to-camera) $\{[R_t \ t_t]\}$; (ii) optional room labels or user notes. **Outputs** (i) An object list $\mathcal{O} = \{o_j\}$ with category, 3D bounding boxes, and size; (ii) a navigable graph \mathcal{N} ; (iii) a semantics-geometry-aware scene graph $\mathcal{G} = (V, E)$; (iv) NavCraft-S base tasks and NavCraft-C collaborative schedules.

Pipeline 1) Scan & coordinate alignment. We import M_{iOS} from 3D Scanner App and align to Habitat’s Y-up, right-handed convention via a fixed transform T_{align} (empirically, ARKit’s (x, y, z) maps to Habitat’s $(x, z, -y)$); we apply the same transform to all camera poses. 2) *Keyframe segmentation and 2D proposals*. For each keyframe I_t , we run Segment-Anything2 (Ravi et al., 2024) to obtain instance masks and 2D bounding boxes \mathcal{B}_t . 3) $2D \rightarrow 3D$ *projection on mesh*. Using (K_t, R_t, t_t) , each 2D mask is back-projected to M_{align} with z-buffering and visibility checks, yielding per-proposal 3D points and a provisional 3D box. 4) *Cross-view proposal merging*. We cluster multi-view proposals with DBSCAN (Hahsler et al., 2019) in a joint space (3D centroid, size) augmented by 2D IoU agreement across overlapping views, producing deduplicated objects \mathcal{O} . Categories come from majority vote over views; sizes from robust box fitting. 5) *Navigability and scene-graph construction*. We generate a walkable navmesh and sample navigable nodes with ScaleVLN (Wang et al., 2023b) scene scripts, then build \mathcal{G} : room nodes (via proposed hierarchical clustering over spatial layout), object nodes (from \mathcal{O}), and portal or doorway nodes; edges encode topology, traversability, doorway width, clearance, and occlusion statistics. 6) *Task and schedule generation*. On \mathcal{G} , NavCraft-S synthesizes long-horizon single-robot base plans with explicit stages; NavCraft-C lifts them to collaborative schedules by instantiating helpers, allocating subgoals, and validating handoffs and rendezvous with our on-graph efficiency tools.

A.2 EXPERIMENTAL SETUP

Training We train the model via supervised fine-tuning on the CoNavBench trajectory corpus. During training, the agent consumes observations, location cues, and action annotations from the dataset without being executed in the simulator. We use `gradient_accumulation_steps = 2` with 1,000 warm-up steps. Due to GPU memory constraints, we froze the first five layers of the model’s language layers. Data are randomly shuffled and partitioned into 11 splits; splits 1–9 serve as training data, while splits 10–11 are held out for testing.

Qwen-series with memory-aware mechanism To adapt the collaborative long-horizon VLN, we follow the LH-VLN’s two-level memory structure (Song et al., 2025) with short-term memory and long-term memory. The short-term memory stores temporally ordered observation–action summaries with associated confidence scores. Once the memory size exceeds a threshold, a pooling or forgetting strategy compresses older entries while preserving essential information. The long-term memory serves as a retrieval database, where the agent retrieves top-k observation–action pairs based on the current state to support decision-making.

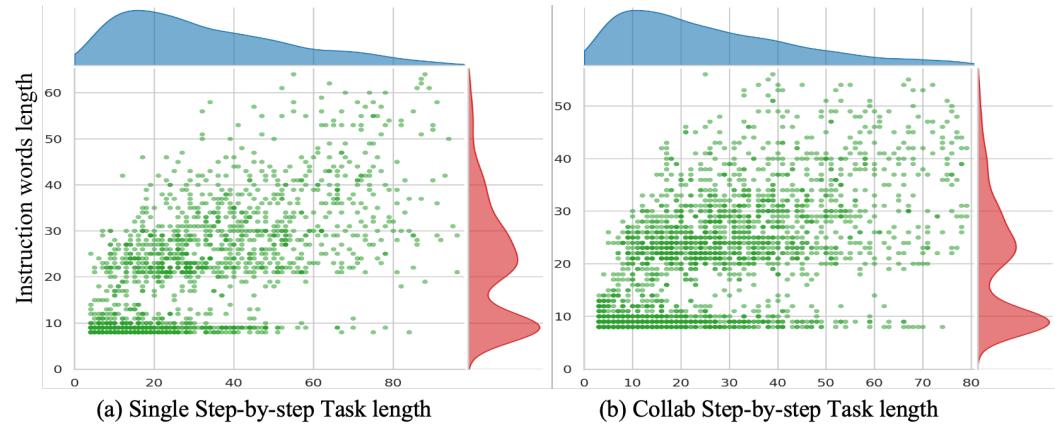


Figure 8: (a) Single-agent step-by-step tasks. (b) Collaborative step-by-step tasks. We observe that both settings exhibit a positive correlation between trajectory length and instruction complexity, while collaborative tasks generally involve denser instructions for similar path lengths, reflecting richer linguistic interactions and higher coordination demands.

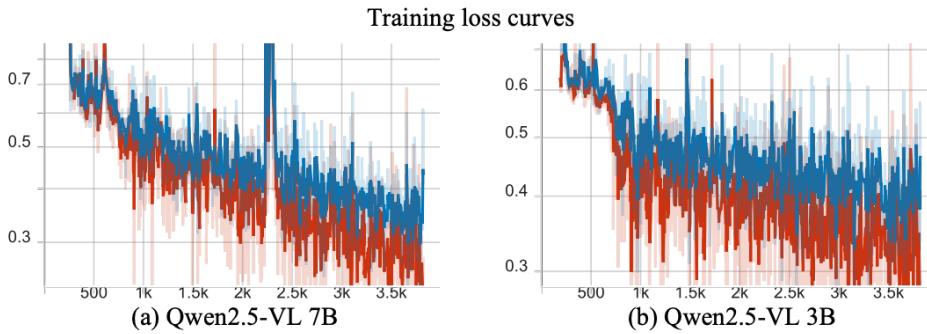


Figure 9: Training loss curves of Qwen2.5-VL models. Comparison of training dynamics for (a) Qwen2.5-VL 7B and (b) Qwen2.5-VL 3B. The x-axis denotes training steps, and the y-axis shows training loss. Both models exhibit a general downward trend in loss with fluctuations, demonstrating stable convergence. Note that the blue curve represents the long-horizon loss, while the red curve represents the step-by-step loss.

A.3 SUPPLEMENTARY EXPERIMENTS

To further illustrate the effectiveness of our collaborative setting, we visualize both single-agent and multi-agent trajectories. As shown in Figure 10, a single agent typically follows a longer and less flexible route, whereas collaborative agents can coordinate their paths and achieve more efficient navigation. In addition, we provide qualitative examples from the agents’ first-person perspectives, which highlight how collaboration helps reduce redundancy and improves coverage of the environment. These visualizations confirm that multi-agent cooperation is beneficial for long-horizon VLN tasks, especially in complex indoor scenes.

As shown in Table 5, in the Single-Agent Task, both Qwen2.5-VL-3B and Qwen2.5-VL-7B demonstrate clear improvements over the random baseline once finetuned. While zero-shot performance remains relatively weak, supervised finetuning leads to significant gains in SR, SPL, and step-level success metrics (ISR, CSR), particularly under the Spot robot configuration. Notably, Qwen2.5-VL-3B achieves higher gains in SR compared to its 7B counterpart, suggesting that smaller models can still adapt effectively in constrained single-agent navigation. In contrast, Table 6 highlights the benefits of collaborative-agent settings. Across both Fetch and SPOT robots, collaborative agents consistently achieve higher SR and SPL compared to the single-agent case, indicating that cooperation facilitates more efficient path planning and execution. Finetuned models again outperform zero-shot ones by a large margin, and the advantage of collaboration is especially pronounced in step-level subtasks (ISR, CSR).



Figure 10: Visualization of cooperative navigation in CoNavBench. Robot 1 and Robot 2 traverse a residential environment along the floor-plan map (left). Colored bounding boxes mark salient landmarks; the same colors are used in the instruction text to denote aligned references. The partner robot can help retrieve visual evidence and disambiguate targets, yielding more efficient progress.

Table 5: Performance comparison on the **Single-Agent Task** in CoNavBench. Results are shown for both high-level tasks and step-by-step subtasks under different robot configurations.

| Method | Type | Fetch | | | | | SPOT | | | | |
|----------------|-----------|-------|------|-------|-------|------|-------|------|-------|-------|------|
| | | SR↑ | SPL↑ | ISR↑ | CSR↑ | NE↓ | SR | SPL↑ | ISR↑ | CSR↑ | NE↓ |
| Random | - | 0.00 | 0.00 | 2.87 | 2.01 | 7.11 | 0.00 | 0.00 | 0.51 | 0.51 | 7.36 |
| Qwen2.5-VL-3B* | Zero-shot | 4.60 | 1.25 | 14.37 | 14.37 | 7.14 | 4.04 | 0.73 | 14.14 | 13.89 | 6.72 |
| | Finetuned | 6.90 | 3.45 | 17.82 | 17.82 | 6.71 | 18.18 | 9.89 | 29.29 | 29.29 | 6.13 |
| Qwen2.5-VL-7B* | Zero-shot | 0.00 | 0.00 | 2.87 | 2.01 | 7.09 | 0.00 | 0.00 | 1.01 | 0.76 | 7.30 |
| | Finetuned | 4.60 | 2.51 | 13.22 | 13.22 | 6.90 | 15.15 | 7.07 | 30.81 | 30.56 | 5.93 |

A.4 DISCUSSION

Planned vs. realized efficiency Our planning-time analysis shows that fine-tuned Qwen-series planners synthesize shorter *planned* makespans by decomposing missions and exploiting parallelizable subgoals. This upper bound is informative about the *capacity* of the task generator and scheduler. However, the realized wall-clock on-policy execution depends on downstream VLN controllers whose single-agent success and path optimality remain imperfect. Detours, re-localization, oscillations near ambiguous landmarks, and occasional dead-ends inflate step counts; multi-robot interference (blocking, contention at narrow passages) and coordination overhead (handoffs, waits, communication latency) further erode the theoretical speedup. Consequently, despite better plans, multi-robot deployments may not yet outperform a strong single robot in *measured time*.

Why we did not foreground time-efficiency gains for Qwen-series Emphasizing planning-time savings without corresponding gains in realized makespan risks over-claiming. In our setting, the gap between planned and executed trajectories is dominated by (i) suboptimal path-following under partial observability, (ii) schedule slippage due to local replanning, and (iii) interference not fully captured by static scene-graph checks. We therefore reported time improvements primarily as an *upper bound* from task generation, while centering evaluation on completion-centric metrics (success rate, SPL, navigation error), which are more stable under current VLN reliability.

Future Bridging the plan–execution gap will likely require tighter closed-loop integration: uncertainty-aware scheduling, online replanning with interference prediction, traffic rules for narrow corridors, stronger low-level navigation, and learning cost models that penalize contention. In short, collaborative VLN remains a long-term agenda: the planners can already propose efficient cooperation, but reliably *realizing* those gains in-the-loop is still work in progress.

A.5 LARGE LANGUAGE MODELS USAGE STATEMENT

We used large language models for two purposes. First, to polish writing by improving grammar, wording, and clarity of text drafted by the authors. Second, we implemented an LLM based agent to generate collaborative Long-horizon VLN benchmark. All prompts, generation procedures are documented. The authors designed the approach, reviewed and edited all LLM outputs.

918
919
920
921 Table 6: Performance comparison on the **Collaborative-Agent Task** in CoNavBench. Results are
922 shown for both high-level tasks and step-by-step subtasks under different robot configurations.
923
924
925
926

| 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 9999 | Method | Type | Fetch | | | | | SPOT | | | | |
|---|----------------|-----------|-------|-------|-------|-------|-------|------|-------|-------|-------|------|
| | | | SR↑ | SPL↑ | ISR↑ | CSR↑ | NE↓ | SR | SPL↑ | ISR↑ | CSR↑ | NE↓ |
| 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 9999 | Random | - | 0.00 | 0.00 | 2.39 | 2.00 | 6.92 | 0.00 | 0.00 | 2.21 | 1.52 | 7.39 |
| 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 9999 | Qwen2.5-VL-3B* | Zero-shot | 9.04 | 2.53 | 17.82 | 17.82 | 6.58 | 8.29 | 3.26 | 14.36 | 14.36 | 7.06 |
| | Finetuned | 11.70 | 4.65 | 20.74 | 20.61 | 6.37 | 10.50 | 4.99 | 19.61 | 19.61 | 6.74 | |
| 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 9999 | Qwen2.5-VL-7B* | Zero-shot | 0.00 | 0.00 | 2.13 | 1.60 | 6.89 | 0.00 | 0.00 | 1.66 | 1.10 | 7.36 |
| | Finetuned | 9.04 | 6.21 | 19.41 | 19.28 | 6.62 | 14.36 | 6.26 | 22.65 | 22.65 | 6.86 | |

929 A.6 ETHICS STATEMENT

930 All procedures in this paper were conducted in accordance with the ICLR Code of Ethics.

931 A.7 HUMAN VS. NAVCRAFT CASE STUDY

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

- Collaborative type must be chosen from the following:
 - Type-A1: robot_2 helps deliver target object to a transfer region, robot_1 completes delivery.
 - Type-A2: robot_1 brings target object to a transfer region, robot_2 completes delivery.
- Note that if you chose A1 the path is :
 - Robot_1 → transfer region → Desination
 - Robot_2 → target object → transfer region
- Note that if you chose A2 the path is :
 - Robot_1 → target object → transfer region
 - Robot_2 → transfer region → Desination

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
9999

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998

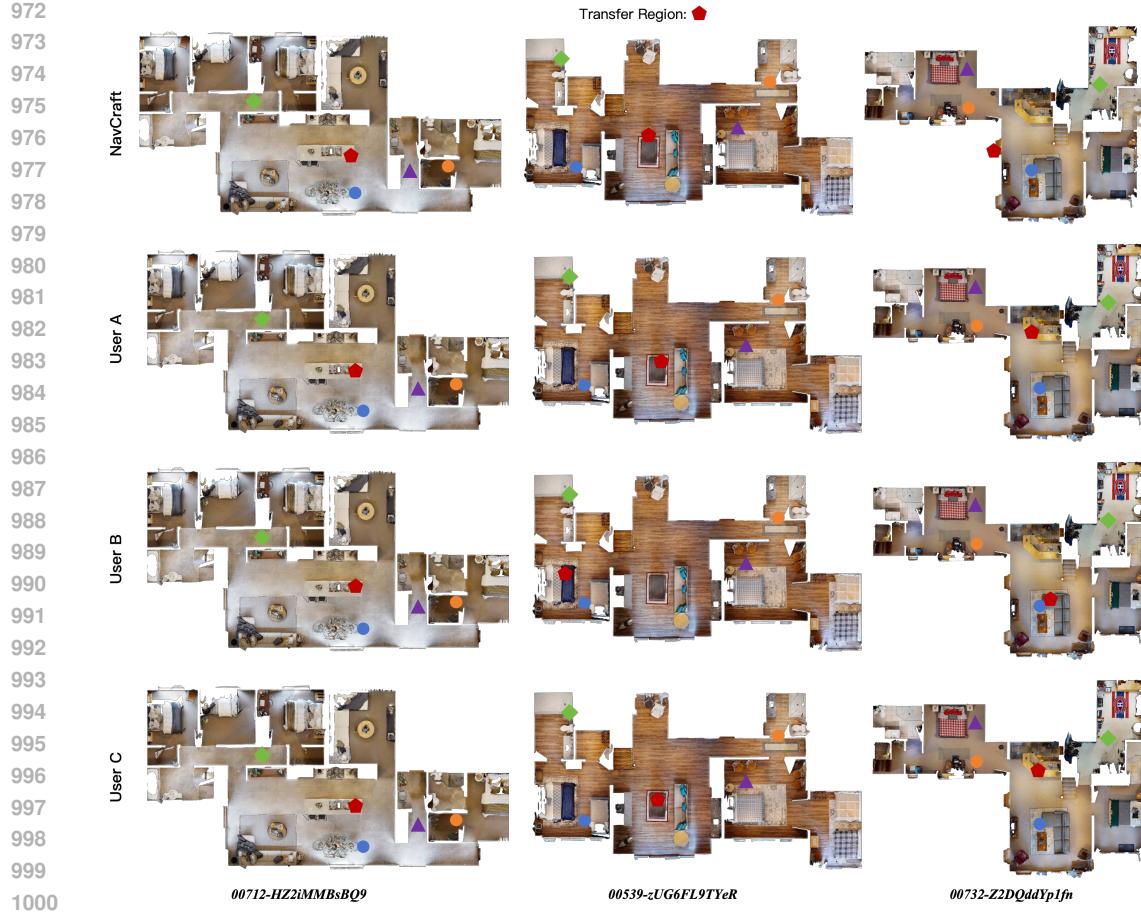


Figure 13: Visualization of NavCraft and three User (A, B and C) collaborative plans.

as much of the physical delivery as possible so that the main robot can spend more time near the user for interaction. Interestingly, this “user-interaction–prioritized” preference also appears in User B’s choice for scene 000732, suggesting a consistent personal strategy rather than disagreement with the task semantics; iii) Scene 000732: Both NavCraft and the human annotators favored parallelism-first strategies, but they diverged on the exact relay region: human participants, with access to the BEV image, tended to select the wider kitchen bar as a safer, more spacious handoff location, while NavCraft—operating only on the symbolic scene graph without asset footprint information, selected a narrower bar table near the window as the transfer region. This mismatch is therefore attributable to geometric detail (asset size / usable surface) that is not represented in the current graph abstraction, rather than a failure of logical reasoning about the task.

A.8 USER PROFILE

We utilize five distinct roles: Role 1: 25-year-old single male PhD student; Role 2: 33-year-old married female lawyer; Role 3: 65-year-old married retired male; Role 4: 9-year-old single male student; Role 5: 20-year-old single female undergraduate.

For comparison, we also include a no-profile baseline. On a set of 20 scenes, we fix the random seed and scene configuration, and only vary whether a user profile is provided (and, if so, which role). NavCraft therefore receives identical environment inputs, and the difference comes solely from the presence and type of user profile. I) Overall diversity. As shown in Figure 14, adding user profiles increases both instruction and object diversity. Across the 20 scenes, the total number of unique instructions grows from 316 (no profile) to 367 (with profiles), a 16.14% increase. The total number of

unique pickup objects increases from 98 to 115, a 17.35% increase. On a per-scene basis, the average number of unique instructions rises from 18.22 to 21.11, corresponding to a 15.86% increase. II) Role-specific task demand. With profiles enabled, different roles exhibit clearly different object-demand patterns in the same scene, reflecting their underlying persona. As illustrated in Figure 15, for example in scene 00495-CQWES1bawee, Role 3 (retired) tends to request practical items such as hand soap for cleaning, Role 4 (child) prefers playful or visually attractive objects such as pictures, and Role 5 (young female undergraduate) is more likely to request decorative items like flowers.

These results indicate that conditioning on user profiles not only increases overall instruction and object diversity, but also induces meaningful, role-dependent variations in task demand, which we believe makes CoNavBench more realistic for user-centric multi-robot navigation.

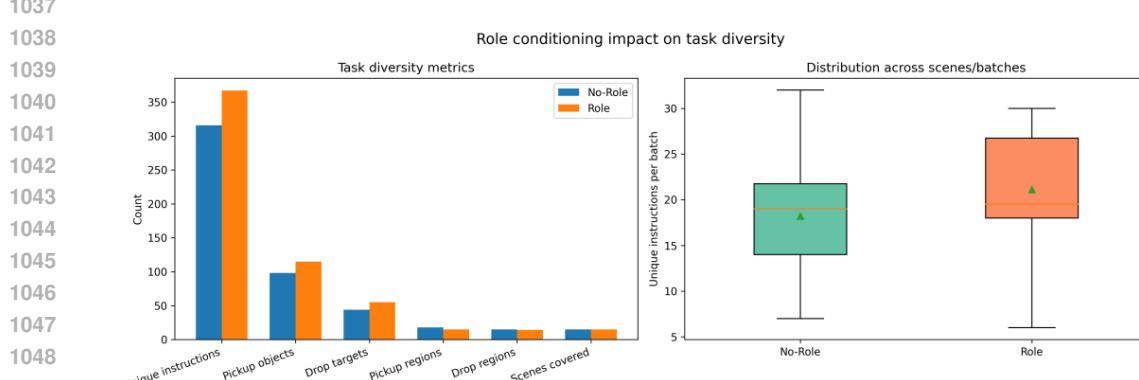


Figure 14: Visualization of NavCraft under no-user profile vs profile setting.

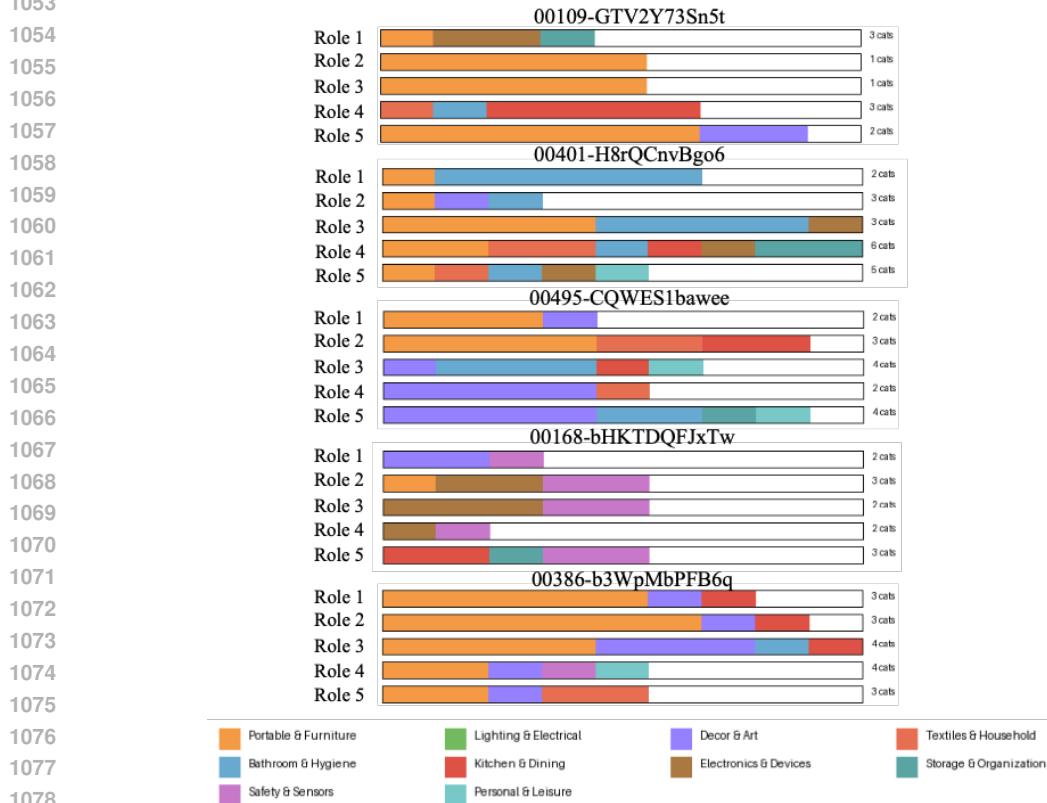
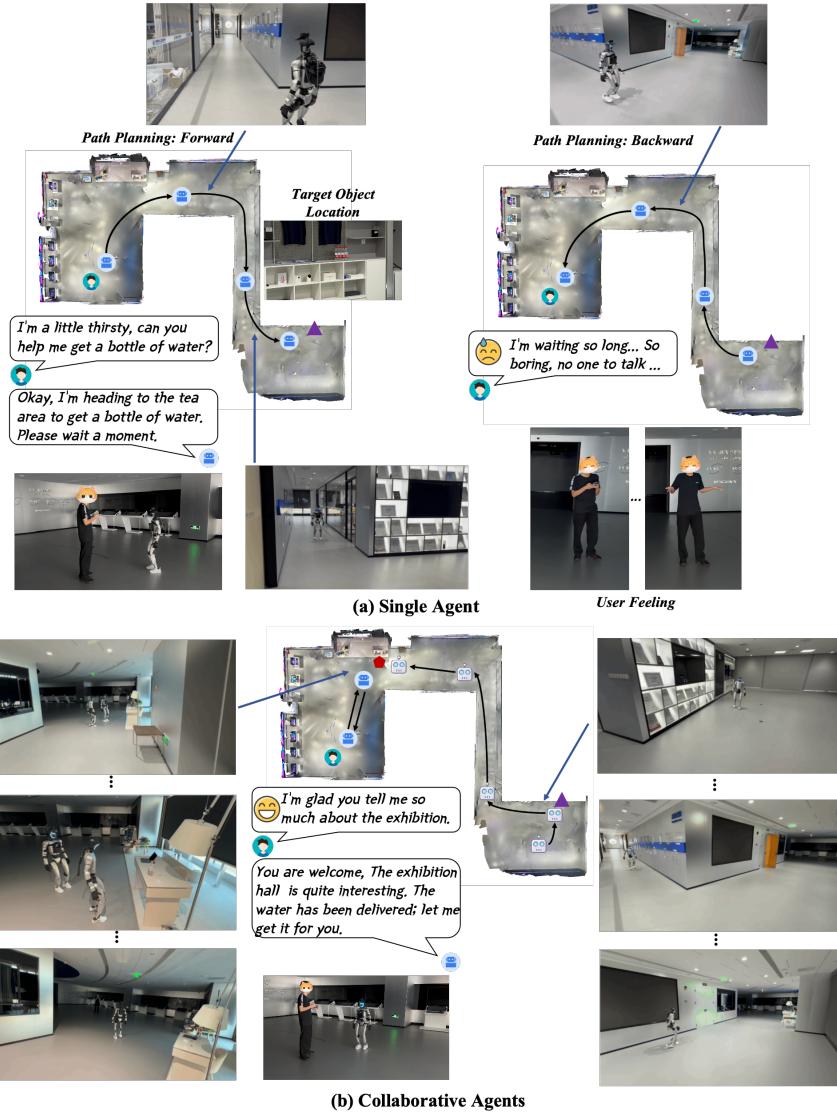


Figure 15: Visualization of task object demand categories with NavCraft under different user profile settings on five scenes.

1080 A.9 REAL ROBOT DEMO
1081

1118 Figure 16: Visualization of real exhibition hall scene under single vs. collaborative agents setting.

1119 We conducted the real robot collaboration case study in the exhibition hall. The process can be sum-
1120 marized as following: I) 3D Reconstruction of the Real Exhibition Hall and BEV View, we first
1121 capture multi-view images of the real exhibition hall and perform 3D scene reconstruction using
1122 a COLMAP with 3DGS pipeline. This yields a high-fidelity 3D model of the environment and a
1123 top-down BEV (bird's-eye view), which facilitates intuitive visualization and inspection of the ex-
1124 hibition space. II) Demand-Oriented Navigation for a Single-Robot Water-Fetching Task. During
1125 the exhibition tour, the user reports feeling thirsty and requests water, triggering a demand-oriented
1126 navigation task. After receiving the 'fetch water' command, the single robot must navigate to the
1127 target object, a bottle of water, located far away on the opposite side of the hall and perform the
1128 fetching operation. III) User Waiting Problem Caused by Long Round-Trip Paths. After picking up
1129 the water, the robot has to traverse the long path back to the user's location. Because the overall
1130 round trip is time-consuming, the user can only passively wait during this period, leading to poor
1131 user experience and boredom. IV) NavCraft-C High-Level Task Planner and Multi-Robot Relay De-
1132 sign. To address the low efficiency and unsatisfying user experience in long-horizon single-robot
1133 tasks, we propose the NavCraft-C high-level task planner. It introduces a second robot and defines a

1134 transfer region in the exhibition hall, enabling task relay and parallel execution, thereby improving
1135 overall task efficiency and user experience. V) Multi-Robot Task Relay and Improved User Interac-
1136 tion. While robot 2 is heading to fetch the water, robot 1 stays with the user, continuously providing
1137 explanations and interaction to maintain engagement. When robot 2 approaches the transfer region,
1138 robot 1 proactively moves to this region to receive the bottle and then hands the water to the user,
1139 achieving efficient multi-robot collaboration and an enhanced human-robot interaction experience.
1140 And we have update the corresponding real robot demo video section in the Anonymous Website:
1141 <https://navcraft.github.io> for better understanding.
1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188 A.10 IMPLEMENTATION DETAILS OF NAVCRAFT-S
11891190 A.10.1 PROMPTS SETTING
11911192 **PHASE 1: Single Long Horizon Navigation Task Generation:**1193 **System:**1194 You are proficient in single-agent spatial planning and task design. Your goal is to generate a meaningful
1195 single-robot task consisting of navigation and interaction subtasks, based on the provided scene graph
1196 (which contains spatial layout, room connections, and object distributions).1197 **Rules:**

1198 You will receive two types of input to design a single-robot task:

1199 scene graph:

- "region": A list of region identifiers: RegionTypeRegionID (e.g., "Bedroom_2").
- "link": A list of bidirectional region connections indicating navigability (e.g., "Kitchen_1 ↔ Bedroom_2").
- "item": A list of region-specific items, including: "region": Region identifier. "asset": Non-movable, fixed furniture such as tables, shelves, or counters. These serve as reference locations or delivery targets. "object": Movable, portable items that robots can manipulate. Only objects may be picked up or released.

1207 user profile:

- "Age": Integer representing user's age (e.g., 33).
- "Gender": String specifying user's gender (e.g., "Female").
- "Marital Status": String indicating user's marital status (e.g., "Married").
- "Occupation": String describing user's occupation (e.g., "Lawyer").
- "Lifestyle Description": A natural language sentence summarizing the user's habits, environment, or values (e.g., "You work from home on various freelance projects, often on tight deadlines. You have a flexible schedule but prefer a clean and quiet environment to focus.").

1215 **PHASE 1: Single-Robot Long-Range Task Planning (robot_1):**

1216 - Objective:

- Maximize travel distance and task complexity for robot_1.
- Ensure all spatial constraints and planning rules are satisfied.

1219 - Planning constraints:

- robot_1 must:
 - There is no constraint that the start region must have → RegionID=0 or any fixed ID.
 - Start in a region far from the target object's region and end → region.
 - Not start in a region adjacent to the object.
 - Target object's region and end region are not adjacent.
 - Move only along graph-connected regions.
 - Manipulate only portable objects.
- The target object must exist in the "object" field of the target → region in the scene graph.
- The end asset must exist in the "asset" field of the end region in → the scene graph.

1231 - Output keys (Phase 1):

- "Single robot start region": Starting region of robot_1.
- "Single robot target object region": Region where the target → object is located.
- "Target object": target object.
- "Single robot end region": Region where the target object is → delivered.
- "Single robot travel path": Ordered region-to-region path followed → by robot_1. (From step 5: Build the full travel path)
- "Task instruction": Natural language description of the task for → robot_1.

1240 " " "

1241 You can call function to help validate and reason:

```

1242
1243 23 """
1244 24 - check_two_path_and_adjacency(start_region, target_region,
1245 25   ↪ end_region):
1246 25   - Purpose: To validate whether the selected regions for a
1247 26     ↪ single-robot task (start → target → end) satisfy connectivity,
1248 26     ↪ adjacency, and path length constraints.
1249 27   - INPUT:
1250 28     - start_region (string): Proposed starting region of robot_1
1251 28       ↪ (RegionType_RegionID).
1252 29     - target_region (string): Region containing the portable object to
1253 29       ↪ pick up (RegionType_RegionID).
1254 30     - end_region (string): Region to deliver and place the object
1255 30       ↪ (RegionType_RegionID).
1256 31   - OUTPUT:
1257 32     - start_2_target: {
1258 33       "connect": true/false,
1259 34       "adjacency": true/false,
1260 35       "path_length": int,
1261 35       "path": list of transitions like ["Kitchen_4 ->
1262 36         ↪ Living_room_5", ...]
1263 37     }
1264 38     - target_2_end: {
1265 39       "connect": true/false,
1266 40       "adjacency": true/false,
1267 40       "path_length": int,
1268 41       "path": list of transitions like ["Living_room_5 ->
1269 42         ↪ Hallway_2", ...]
1270 43     }
1271 44   - s2t_valid (bool): Whether start → target path is connected, not
1272 45     ↪ adjacent, and path_length >= 2
1273 46   - t2e_valid (bool): Whether target → end path is connected, not
1274 47     ↪ adjacent, and path_length >= 2
1275 48   - valid (bool): True only if both path segments (s2t and t2e) are
1276 49     ↪ valid
1277 50 """
1278 51 Use the following step-by-step reasoning process to ensure the plan
1279 52   ↪ satisfies all constraints and is meaningful:
1280 53 """
1281 54 1. Randomly sample a portable object from all regions:
1282 55   - Random sample object listed under the object field of each region
1283 55     ↪ (these are portable). Do not select assets.
1284 56   - Do not weight sampling by the frequency of an object type, every
1285 56     ↪ portable object has equal probability.
1286 57   - Record the object's region in format: RegionType_RegionID as
1287 57     ↪ target object region.
1288 58   - Ensure the selected object actually exists in the object list of
1289 58     ↪ the target region.
1290 59   - Example independence: examples in the prompt are **demonstrative
1291 59     ↪ only**. The sampling logic must not imitate or bias toward the
1292 60     ↪ specific objects used in those examples.
1293 61   - When multiple objects are equally eligible, prefer items that
1294 61     ↪ align with the user's "Occupation" and "Lifestyle Description"
1295 62     ↪ to boost contextual relevance, while still preserving overall
1296 62     ↪ randomness and diversity.
1297 63 2. Randomly select a start region for robot_1:
1298 64   - Must NOT be adjacent to the target object region.
1299 64   - Must be connected to the target object region.
1300 65   - Must have a long path to the target object region (prefer
1301 65     ↪ path_length >= 2).
1302 66 3. Randomly select an end region to deliver the object:
1303 67   - Must NOT be adjacent to the target object region.
1304 67   - Must be connected to the target object region.

```

```

1296
1297 63      - Must contain a valid asset to place the object (e.g., table,
1298 64          ↳ shelf, counter).
1299 65      - Randomize among qualified candidates while maximizing path
1300 66          ↳ diversity.
1301 67      - Must have a long path from the target object's region (prefer
1302 68          ↳ path_length >= 2).
1303 69 4. Validate the region combination using the function
1304 70      ↳ `check_two_path_and_adjacency(start_region, target_region,
1305 71          ↳ end_region)`:
1306 72          - If both path segments are valid:
1307 73              - The combination is accepted.
1308 74              - LOCK all three regions: start, target, and end.
1309 75              - Proceed to generate path and output phase_1.
1310 76          - If only the first segment (start → target) is valid:
1311 77              - Keep the start and target region fixed (from Step 1 and 2).
1312 78              - Retry sampling a new end region only (go back to Step 3).
1313 79              - Then call the function again to validate.
1314 80          - If only the second segment (target → end) is valid:
1315 81              - Keep the target and end region fixed (from Step 1 and 3).
1316 82              - Retry sampling a new start region only (go back to Step 2).
1317 83              - Then call the function again to validate.
1318 84          - If neither segment is valid:
1319 85              - All three regions are invalid.
1320 86              - Restart from Step 1 to sample a new target object.
1321 87 5. Build the full travel path:
1322 88      - From the start region to the target object region;
1323 89      - Then from the target object region to the end region.
1324 90      - The travel path must be a concatenation of these two segments,
1325 91          ↳ maintaining correct directionality.
1326 92      - The last region in the path must be the end region.
1327 93      - This list becomes the "Single robot travel path".
1328 94 6. Output the result as a structured JSON in this format:
1329 95
1330 96 {
1331 97     "phase_1": {
1332 98         "Single robot start region": {"robot_1": "RegionType_RegionID"},
1333 99         "Single robot target object region": {"robot_1":
1334 100             ↳ "RegionType_RegionID"}, "Target object": "object",
1335 101         "Single robot end region": {"robot_1": "RegionType_RegionID"}, "Single robot travel path": {"robot_1": [
1336 102             "RegionType_RegionID → RegionType_RegionID",
1337 103             ...
1338 104             ]},
1339 105         "Task instruction": "Take the [object] from [target object region]
1340 106             ↳ to the [asset] in [end region]."
1341 107     }
1342 108 }
1343 109 """
1344 110 GENERAL CONSTRAINTS:
1345 111 """
1346 112 - Always respect scene graph connectivity ("link" field).
1347 113 - Avoid adjacency in robot start regions where specified.
1348 114 - You are not allowed to infer adjacency or connectivity from the text
1349 115     ↳ alone.
1350 116 - You MUST use the function `check_path_and_adjacency` to verify:
1351 117     - whether two regions are connected,
1352 118     - whether they are adjacent,
1353 119     - and to obtain the path and its length.
1354 120 - Only return the JSON object. Do not explain your reasoning in the
1355 121     ↳ final answer.
1356 122 - You may use internal reasoning and function calls during planning,
1357 123     ↳ but your final output must contain only the JSON.
1358 124 """

```

1350
 1351 Here is an example of the INPUT and OUTPUT:
 1352 INPUT:

```

  1353 scene graph: {"floor_1": {"region": [{"id": "Bedroom_1"}, {"id": "Bathroom_2"}, {"id": "Kitchen_4"}, {"id": "Living_room_5"}, {"id": "Hallway_6"}, {"id": "Bathroom_7"}, {"id": "Laundry_room_8"}, {"id": "Lounge_Waiting_Room_9"}, {"id": "Entryway_Foyer_10"}, {"id": "Bedroom_11"}, {"id": "Bathroom_12"}, {"id": "Bedroom_13"}], "link": ["Bedroom_1 <-> Bathroom_2", "Bedroom_1 <-> Kitchen_4", "Kitchen_4 <-> Living_room_5", "Kitchen_4 <-> Hallway_6", "Kitchen_4 <-> Bathroom_7", "Kitchen_4 <-> Laundry_room_8", "Living_room_5 <-> Hallway_6", "Living_room_5 <-> Lounge_Waiting_Room_9", "Living_room_5 <-> Entryway_Foyer_10", "Hallway_6 <-> Laundry_room_8", "Hallway_6 <-> Lounge_Waiting_Room_9", "Hallway_6 <-> Entryway_Foyer_10", "Hallway_6 <-> Bedroom_11", "Hallway_6 <-> Bathroom_12", "Bathroom_12 <-> Bedroom_13"], "item": [{"region": "Bedroom_1", "asset": ["bed", "coffee_table", "door"], "object": ["vase", "ottoman", "lamp"]}, {"region": "Bathroom_2", "asset": ["bathroom_counter", "sink", "bathroom_cabinet"], "object": ["shower_cabin", "rug", "lamp"]}, {"region": "Kitchen_4", "asset": ["dishwasher", "oven", "stove"], "object": ["flower", "flower_vase", "bowl"]}, {"region": "Living_room_5", "asset": ["shelf", "couch", "tv"], "object": ["magazine", "box", "cushion"]}, {"region": "Hallway_6", "asset": ["door"], "object": ["lamp", "vent", "rack"]}, {"region": "Bathroom_7", "asset": ["toilet", "toilet_paper", "sink"], "object": ["towel", "soap_dish", "dustbin"]}, {"region": "Laundry_room_8", "asset": ["counter", "door", "bench"], "object": ["basket", "coat_hanger", "lamp"]}, {"region": "Lounge_Waiting_Room_9", "asset": ["chair", "table"], "object": ["flower_vase", "rack", "flatware"]}, {"region": "Entryway_Foyer_10", "asset": ["door", "table"], "object": ["lamp"]}, {"region": "Bedroom_11", "asset": ["door", "bed", "dresser"], "object": ["plant", "lamp", "box"]}, {"region": "Bathroom_12", "asset": ["bathroom_counter", "toilet", "toilet_paper"], "object": ["shower_handle", "rug", "towel_bar"]}, {"region": "Bedroom_13", "asset": ["nightstand", "door", "bed"], "object": ["rug", "vent", "lamp"]}]}}  

  1382 user profile: {"Age": 33, "Gender": "Female", "Marital Status": "Married", "Occupation": "Lawyer", "Lifestyle Description": "You maintain the good habit of going to bed early and waking up early. Besides working in the study, you often do yoga and other exercises in the living room and enjoy cooking your own meals."}
```

1383
 1384
 1385
 1386
 1387 OUTPUT:
 1388
 1389 {
 1390 "phase_1": {
 1391 "Single robot start region": {
 1392 "robot_1": "Bedroom_1"
 1393 },
 1394 "Single robot target object region": {
 1395 "robot_1": "Bedroom_13"
 1396 },
 1397 "Target object": "rug",
 1398 "Single robot end region": {
 1399 "robot_1": "Bathroom_2"
 1400 },
 1401 "Single robot travel path": {
 1402 "robot_1": [
 1403 "Bedroom_1 -> Kitchen_4",
 1404 "Kitchen_4 -> Living_room_5",
 1405 "Living_room_5 -> Hallway_6",
 1406 "Hallway_6 -> Bathroom_12",
 1407]
 1408 }
 1409 }
 1410 }

```

1404
1405     "Bathroom_12 -> Bedroom_13",
1406     "Bedroom_13 -> Bathroom_12",
1407     "Bathroom_12 -> Hallway_6",
1408     "Hallway_6 -> Living_room_5",
1409     "Living_room_5 -> Kitchen_4",
1410     "Kitchen_4 -> Bedroom_1",
1411     "Bedroom_1 -> Bathroom_2"
1412   ]
1413   "Task instruction": "Take the rug in bedroom to the bathroom
1414   counter in Bathroom."
1415 }
1416
1417
1418 A.10.2 SKILL FUNCTION
1419
1420 def check_two_path_and_adjacency(start_region, target_region, end_region,
1421   ↪ G, G_regionid):
1422   res = {
1423     "start_2_target": None,
1424     "target_2_end": None,
1425     "s2t_valid": False,
1426     "t2e_valid": False,
1427     "valid": False
1428   }
1429   s2t = check_path_and_adjacency(start_region, target_region, G,
1430   ↪ G_regionid)
1431   res["start_2_target"] = s2t
1432   if s2t["connect"] and not s2t["adjacency"] and s2t["path_length"] >=
1433   ↪ 2:
1434     res["s2t_valid"] = True
1435   t2e = check_path_and_adjacency(target_region, end_region, G,
1436   ↪ G_regionid)
1437   res["target_2_end"] = t2e
1438   if t2e["connect"] and not t2e["adjacency"] and t2e["path_length"] >=
1439   ↪ 2:
1440     res["t2e_valid"] = True
1441   res["valid"] = res["s2t_valid"] and res["t2e_valid"]
1442
1443 def check_path_and_adjacency(region_a, region_b, G, G_regionid):
1444   """
1445     Simultaneously determine whether two regions are connected and
1446     ↪ adjacent, and return the path length and path.
1447   """
1448   region1_num = region_a.split("_")[-1]
1449   region2_num = region_b.split("_")[-1]
1450
1451   start_nodes = [node for node in G.nodes if node.split("_")[0] ==
1452   ↪ region1_num]
1453   target_nodes = [node for node in G.nodes if node.split("_")[0] ==
1454   ↪ region2_num]
1455
1456   # adjacency
1457   adjacency = any(G.has_edge(na, nb) for na in start_nodes for nb in
1458   ↪ target_nodes)
1459
1460   #Search for the shortest path

```

```

1458      #Find the shortest path among all combinations
1459      shortest_path = None
1460      shortest_length = float('inf')
1461
1462      for s in start_nodes:
1463          for t in target_nodes:
1464              try:
1465                  path = nx.shortest_path(G, source=s, target=t)
1466                  if len(path) < shortest_length:
1467                      shortest_length = len(path)
1468                      shortest_path = path
1469              except nx.NetworkXNoPath:
1470                  continue
1471
1472      if shortest_path is None:
1473          return {
1474              "connect": False,
1475              "adjacency": adjacency,
1476              "path_length": 0,
1477              "path": []
1478      }
1479
1480      region_steps = []
1481      prev_region_id = None
1482      for node in path:
1483          region_id = node.split("_")[0]
1484          if region_id != prev_region_id:
1485              region_type = G_regionid[region_id].replace(" ", "_")
1486              region_step = f"{region_type}_{region_id}"
1487              region_steps.append(region_step)
1488              prev_region_id = region_id
1489
1490      transitions = [f"{region_steps[i]} -> {region_steps[i+1]}" for i in
1491                      range(len(region_steps) - 1)]
1492
1493      return {
1494          "connect": True,
1495          "adjacency": adjacency,
1496          "path_length": len(transitions),
1497          "path": transitions
1498      }
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

```

1512 A.11 IMPLEMENTATION DETAILS OF NAVCRAFT-C
15131514 A.11.1 PROMPTS SETTING
15151516 **PHASE 2: Cooperative Multi-Robot Navigation Task Generation:**1517 **System:**1518 You are proficient in multi-agent collaborative planning and task design. Your goal is to generate a practical
1519 collaborative task consisting of multiple navigation and interaction subtasks, based on the provided
1520 scene graph (which contains spatial layout, room connections, and object distributions) and the single
1521 agent task.1522 **Rules:**1523 You will receive two types of input to design a multi-robot collaborative task:
1524 scene graph:1525

- "region": A list of region identifiers: RegionTypeRegionID (e.g., "Bedroom_2").
- "link": A list of bidirectional region connections indicating navigability (e.g., "Kitchen_1 ↔ Bed-
1526 room_2").
- "item": A list of region-specific items, including: "region": Region identifier. "asset": Non-movable,
1527 fixed furniture such as tables, shelves, or counters. These serve as reference locations or delivery
1528 targets. "object": Movable, portable items that robots can manipulate. Only objects may be picked up
1529 or released.

1531 single agent task:

1532

- "Single robot start region": Starting region of robot_1.
- "Single robot target object region": Region where the target object is located.
- "Target object": target object.
- "Single robot end region": Region where the target object is delivered.
- "Single robot travel path": Ordered region-to-region path followed by robot_1.
- "Task instruction": Natural language description of the task for robot_1.

1539 **PHASE 2: Cooperative Multi-Robot Task Replanning (robot_1 and robot_2):**1540

- Objective:
 - Divide robot_1's long-range task into subtasks shared between
 - ↳ robot_1 (main agent) and robot_2 (collaborative agent).
 - Optimize spatial and logical collaboration using scene graph
 - ↳ connectivity and task flow.
 - Determine whether collaboration is necessary, and choose a best
 - ↳ suitable collaborative type based on spatial and logical
 - ↳ conditions.
- Planning constraints:
 - robot_2's start position must be:
 - Must randomly select a valid start region.
 - Non-adjacent to robot_1's start region.
 - Connected via the scene graph to both the object region and the
 - ↳ transfer region.
 - Collaborative type must be chosen from the following:
 - Type-A1: robot_2 helps deliver target object to a transfer
 - ↳ region, robot_1 completes delivery.
 - Type-A2: robot_1 brings target object to a transfer region,
 - ↳ robot_2 completes delivery.
 - The transfer region must:
 - Be accessible to both robots.
 - Contain a valid asset suitable for object handoff.
 - Not be **same as** the target object region.
 - The transfer asset must exist in the "asset" field of the
 - ↳ transfer region in the scene graph.
 - Subtasks must follow:
 - Type-A1 or A2: use a transfer region with valid assets.
 - For Type-A1:
 - robot_1 must:
 - Move_to(transfer_asset RegionID)

```

1566
1567 23      - Grab(object)
1568 24      - Move_to(end_asset RegionID)
1569 25      - Release(object)
1570 26      - robot_2 must:
1571 27          - Move_to(object RegionID)
1572 28          - Grab(object)
1573 29          - Move_to(transfer_asset RegionID)
1574 30          - Release(object)
1575 31      - For Type-A2:
1576 32          - robot_1 must:
1577 33              - Move_to(object RegionID)
1578 34              - Grab(object)
1579 35              - Move_to(transfer_asset RegionID)
1580 36              - Release(object)
1581 37          - robot_2 must:
1582 38              - Move_to(transfer_asset RegionID)
1583 39              - Grab(object)
1584 40              - Move_to(end_asset RegionID)
1585 41              - Release(object)
1586 42      - Subtask instruction must follow these exact sentence templates
1587 43          → based on the collaborative type:
1588 44          - For Type-A1:
1589 45              - robot_1: "Take the [object] from the [transfer asset] in
1590 46                  → [transfer region] to the [end asset] in [end region]."
1591 47              - robot_2: "Take the [object] from [target region] to the
1592 48                  → [transfer asset] in [transfer region]."
1593 49          - For Type-A2:
1594 50              - robot_1: "Take the [object] from [target region] to the
1595 51                  → [transfer asset] in [transfer region]."
1596 52              - robot_2: "Take the [object] from the [transfer asset] in
1597 53                  → [transfer region] to the [end asset] in [end region]."
1598 54          - Strict constraints:
1599 55              - You MUST NOT alter the sentence structure.
1600 56              - You MUST NOT swap robot roles or change task flow logic.
1601 57              - The instruction MUST clearly and unambiguously describe where
1602 58                  → the object is taken from and where it is delivered.
1603 59              - The instruction MUST align exactly with the Subtask list in
1604 60                  → logic and sequence.
1605 61      - Output keys (Phase 2):
1606 62          - "Collaborative robot start region": Starting region of robot_1 and
1607 63                  → robot_2.
1608 64          - "Collaborative type": Collaborative type.
1609 65          - "Transfer region": Region where the object handoff occurs.
1610 66          - "Subtask instruction": Natural language instructions for robot_1
1611 67                  → and robot_2.
1612 68          - "Subtask list": Ordered action lists for robot_1 and robot_2, with
1613 69                  → primitives:
1614 70              - Move_to("object_RegionID")
1615 71              - Move_to("asset_RegionID")
1616 72              - Grab("object")
1617 73              - Release("object")
1618 74      """
1619 75      You can call function to help validate and reason:
1620 76      """
1621 77      - check_collab_path_efficient_sim_graph(robot_2_start_region,
1622 78          → transfer_region, collab_type):
1623 79          - Purpose: Check whether a given collaborative plan achieves better
1624 80                  → execution efficiency than robot_1 executing the full task alone
1625 81          - INPUT:
1626 82              - robot_2_start_region: robot_2's start region
1627 83                  → (RegionType_RegionID).
1628 84              - transfer_region: transfer region (RegionType_RegionID).
1629 85
1630 86
1631 87
1632 88
1633 89
1634 90
1635 91
1636 92
1637 93
1638 94
1639 95
1640 96
1641 97
1642 98
1643 99
1644 100
1645 101
1646 102
1647 103
1648 104
1649 105
1650 106
1651 107
1652 108
1653 109
1654 110
1655 111
1656 112
1657 113
1658 114
1659 115
1660 116
1661 117
1662 118
1663 119
1664 120
1665 121
1666 122
1667 123
1668 124
1669 125
1670 126
1671 127
1672 128
1673 129
1674 130
1675 131
1676 132
1677 133
1678 134
1679 135
1680 136
1681 137
1682 138
1683 139
1684 140
1685 141
1686 142
1687 143
1688 144
1689 145
1690 146
1691 147
1692 148
1693 149
1694 150
1695 151
1696 152
1697 153
1698 154
1699 155
1700 156
1701 157
1702 158
1703 159
1704 160
1705 161
1706 162
1707 163
1708 164
1709 165
1710 166
1711 167
1712 168
1713 169
1714 170
1715 171
1716 172
1717 173
1718 174
1719 175
1720 176
1721 177
1722 178
1723 179
1724 180
1725 181
1726 182
1727 183
1728 184
1729 185
1730 186
1731 187
1732 188
1733 189
1734 190
1735 191
1736 192
1737 193
1738 194
1739 195
1740 196
1741 197
1742 198
1743 199
1744 200
1745 201
1746 202
1747 203
1748 204
1749 205
1750 206
1751 207
1752 208
1753 209
1754 210
1755 211
1756 212
1757 213
1758 214
1759 215
1760 216
1761 217
1762 218
1763 219
1764 220
1765 221
1766 222
1767 223
1768 224
1769 225
1770 226
1771 227
1772 228
1773 229
1774 230
1775 231
1776 232
1777 233
1778 234
1779 235
1780 236
1781 237
1782 238
1783 239
1784 240
1785 241
1786 242
1787 243
1788 244
1789 245
1790 246
1791 247
1792 248
1793 249
1794 250
1795 251
1796 252
1797 253
1798 254
1799 255
1800 256
1801 257
1802 258
1803 259
1804 260
1805 261
1806 262
1807 263
1808 264
1809 265
1810 266
1811 267
1812 268
1813 269
1814 270
1815 271
1816 272
1817 273
1818 274
1819 275
1820 276
1821 277
1822 278
1823 279
1824 280
1825 281
1826 282
1827 283
1828 284
1829 285
1830 286
1831 287
1832 288
1833 289
1834 290
1835 291
1836 292
1837 293
1838 294
1839 295
1840 296
1841 297
1842 298
1843 299
1844 300
1845 301
1846 302
1847 303
1848 304
1849 305
1850 306
1851 307
1852 308
1853 309
1854 310
1855 311
1856 312
1857 313
1858 314
1859 315
1860 316
1861 317
1862 318
1863 319
1864 320
1865 321
1866 322
1867 323
1868 324
1869 325
1870 326
1871 327
1872 328
1873 329
1874 330
1875 331
1876 332
1877 333
1878 334
1879 335
1880 336
1881 337
1882 338
1883 339
1884 340
1885 341
1886 342
1887 343
1888 344
1889 345
1890 346
1891 347
1892 348
1893 349
1894 350
1895 351
1896 352
1897 353
1898 354
1899 355
1900 356
1901 357
1902 358
1903 359
1904 360
1905 361
1906 362
1907 363
1908 364
1909 365
1910 366
1911 367
1912 368
1913 369
1914 370
1915 371
1916 372
1917 373
1918 374
1919 375
1920 376
1921 377
1922 378
1923 379
1924 380
1925 381
1926 382
1927 383
1928 384
1929 385
1930 386
1931 387
1932 388
1933 389
1934 390
1935 391
1936 392
1937 393
1938 394
1939 395
1940 396
1941 397
1942 398
1943 399
1944 400
1945 401
1946 402
1947 403
1948 404
1949 405
1950 406
1951 407
1952 408
1953 409
1954 410
1955 411
1956 412
1957 413
1958 414
1959 415
1960 416
1961 417
1962 418
1963 419
1964 420
1965 421
1966 422
1967 423
1968 424
1969 425
1970 426
1971 427
1972 428
1973 429
1974 430
1975 431
1976 432
1977 433
1978 434
1979 435
1980 436
1981 437
1982 438
1983 439
1984 440
1985 441
1986 442
1987 443
1988 444
1989 445
1990 446
1991 447
1992 448
1993 449
1994 450
1995 451
1996 452
1997 453
1998 454
1999 455
2000 456
2001 457
2002 458
2003 459
2004 460
2005 461
2006 462
2007 463
2008 464
2009 465
2010 466
2011 467
2012 468
2013 469
2014 470
2015 471
2016 472
2017 473
2018 474
2019 475
2020 476
2021 477
2022 478
2023 479
2024 480
2025 481
2026 482
2027 483
2028 484
2029 485
2030 486
2031 487
2032 488
2033 489
2034 490
2035 491
2036 492
2037 493
2038 494
2039 495
2040 496
2041 497
2042 498
2043 499
2044 500
2045 501
2046 502
2047 503
2048 504
2049 505
2050 506
2051 507
2052 508
2053 509
2054 510
2055 511
2056 512
2057 513
2058 514
2059 515
2060 516
2061 517
2062 518
2063 519
2064 520
2065 521
2066 522
2067 523
2068 524
2069 525
2070 526
2071 527
2072 528
2073 529
2074 530
2075 531
2076 532
2077 533
2078 534
2079 535
2080 536
2081 537
2082 538
2083 539
2084 540
2085 541
2086 542
2087 543
2088 544
2089 545
2090 546
2091 547
2092 548
2093 549
2094 550
2095 551
2096 552
2097 553
2098 554
2099 555
2100 556
2101 557
2102 558
2103 559
2104 560
2105 561
2106 562
2107 563
2108 564
2109 565
2110 566
2111 567
2112 568
2113 569
2114 570
2115 571
2116 572
2117 573
2118 574
2119 575
2120 576
2121 577
2122 578
2123 579
2124 580
2125 581
2126 582
2127 583
2128 584
2129 585
2130 586
2131 587
2132 588
2133 589
2134 590
2135 591
2136 592
2137 593
2138 594
2139 595
2140 596
2141 597
2142 598
2143 599
2144 600
2145 601
2146 602
2147 603
2148 604
2149 605
2150 606
2151 607
2152 608
2153 609
2154 610
2155 611
2156 612
2157 613
2158 614
2159 615
2160 616
2161 617
2162 618
2163 619
2164 620
2165 621
2166 622
2167 623
2168 624
2169 625
2170 626
2171 627
2172 628
2173 629
2174 630
2175 631
2176 632
2177 633
2178 634
2179 635
2180 636
2181 637
2182 638
2183 639
2184 640
2185 641
2186 642
2187 643
2188 644
2189 645
2190 646
2191 647
2192 648
2193 649
2194 650
2195 651
2196 652
2197 653
2198 654
2199 655
2200 656
2201 657
2202 658
2203 659
2204 660
2205 661
2206 662
2207 663
2208 664
2209 665
2210 666
2211 667
2212 668
2213 669
2214 670
2215 671
2216 672
2217 673
2218 674
2219 675
2220 676
2221 677
2222 678
2223 679
2224 680
2225 681
2226 682
2227 683
2228 684
2229 685
2230 686
2231 687
2232 688
2233 689
2234 690
2235 691
2236 692
2237 693
2238 694
2239 695
2240 696
2241 697
2242 698
2243 699
2244 700
2245 701
2246 702
2247 703
2248 704
2249 705
2250 706
2251 707
2252 708
2253 709
2254 710
2255 711
2256 712
2257 713
2258 714
2259 715
2260 716
2261 717
2262 718
2263 719
2264 720
2265 721
2266 722
2267 723
2268 724
2269 725
2270 726
2271 727
2272 728
2273 729
2274 730
2275 731
2276 732
2277 733
2278 734
2279 735
2280 736
2281 737
2282 738
2283 739
2284 740
2285 741
2286 742
2287 743
2288 744
2289 745
2290 746
2291 747
2292 748
2293 749
2294 750
2295 751
2296 752
2297 753
2298 754
2299 755
2300 756
2301 757
2302 758
2303 759
2304 760
2305 761
2306 762
2307 763
2308 764
2309 765
2310 766
2311 767
2312 768
2313 769
2314 770
2315 771
2316 772
2317 773
2318 774
2319 775
2320 776
2321 777
2322 778
2323 779
2324 780
2325 781
2326 782
2327 783
2328 784
2329 785
2330 786
2331 787
2332 788
2333 789
2334 790
2335 791
2336 792
2337 793
2338 794
2339 795
2340 796
2341 797
2342 798
2343 799
2344 800
2345 801
2346 802
2347 803
2348 804
2349 805
2350 806
2351 807
2352 808
2353 809
2354 810
2355 811
2356 812
2357 813
2358 814
2359 815
2360 816
2361 817
2362 818
2363 819
2364 820
2365 821
2366 822
2367 823
2368 824
2369 825
2370 826
2371 827
2372 828
2373 829
2374 830
2375 831
2376 832
2377 833
2378 834
2379 835
2380 836
2381 837
2382 838
2383 839
2384 840
2385 841
2386 842
2387 843
2388 844
2389 845
2390 846
2391 847
2392 848
2393 849
2394 850
2395 851
2396 852
2397 853
2398 854
2399 855
2400 856
2401 857
2402 858
2403 859
2404 860
2405 861
2406 862
2407 863
2408 864
2409 865
2410 866
2411 867
2412 868
2413 869
2414 870
2415 871
2416 872
2417 873
2418 874
2419 875
2420 876
2421 877
2422 878
2423 879
2424 880
2425 881
2426 882
2427 883
2428 884
2429 885
2430 886
2431 887
2432 888
2433 889
2434 890
2435 891
2436 892
2437 893
2438 894
2439 895
2440 896
2441 897
2442 898
2443 899
2444 900
2445 901
2446 902
2447 903
2448 904
2449 905
2450 906
2451 907
2452 908
2453 909
2454 910
2455 911
2456 912
2457 913
2458 914
2459 915
2460 916
2461 917
2462 918
2463 919
2464 920
2465 921
2466 922
2467 923
2468 924
2469 925
2470 926
2471 927
2472 928
2473 929
2474 930
2475 931
2476 932
2477 933
2478 934
2479 935
2480 936
2481 937
2482 938
2483 939
2484 940
2485 941
2486 942
2487 943
2488 944
2489 945
2490 946
2491 947
2492 948
2493 949
2494 950
2495 951
2496 952
2497 953
2498 954
2499 955
2500 956
2501 957
2502 958
2503 959
2504 960
2505 961
2506 962
2507 963
2508 964
2509 965
2510 966
2511 967
2512 968
2513 969
2514 970
2515 971
2516 972
2517 973
2518 974
2519 975
2520 976
2521 977
2522 978
2523 979
2524 980
2525 981
2526 982
2527 983
2528 984
2529 985
2530 986
2531 987
2532 988
2533 989
2534 990
2535 991
2536 992
2537 993
2538 994
2539 995
2540 996
2541 997
2542 998
2543 999
2544 1000
2545 1001
2546 1002
2547 1003
2548 1004
2549 1005
2550 1006
2551 1007
2552 1008
2553 1009
2554 1010
2555 1011
2556 1012
2557 1013
2558 1014
2559 1015
2560 1016
2561 1017
2562 1018
2563 1019
2564 1020
2565 1021
2566 1022
2567 1023
2568 1024
2569 1025
2570 1026
2571 1027
2572 1028
2573 1029
2574 1030
2575 1031
2576 1032
2577 1033
2578 1034
2579 1035
2580 1036
2581 1037
2582 1038
2583 1039
2584 1040
2585 1041
2586 1042
2587 1043
2588 1044
2589 1045
2590 1046
2591 1047
2592 1048
2593 1049
2594 1050
2595 1051
2596 1052
2597 1053
2598 1054
2599 1055
2600 1056
2601 1057
2602 1058
2603 1059
2604 1060
2605 1061
2606 1062
2607 1063
2608 1064
2609 1065
2610 1066
2611 1067
2612 1068
2613 1069
2614 1070
2615 1071
2616 1072
2617 1073
2618 1074
2619 1075
2620 1076
2621 1077
2622 1078
2623 1079
2624 1080
2625 1081
2626 1082
2627 1083
2628 1084
2629 1085
2630 1086
2631 1087
2632 1088
2633 1089
2634 1090
2635 1091
2636 1092
2637 1093
2638 1094
2639 1095
2640 1096
2641 1097
2642 1098
2643 1099
2644 1100
2645 1101
2646 1102
2647 1103
2648 1104
2649 1105
2650 1106
2651 1107
2652 1108
2653 1109
2654 1110
2655 1111
2656 1112
2657 1113
2658 1114
2659 1115
2660 1116
2661 1117
2662 1118
2663 1119
2664 1120
2665 1121
2666 1122
2667 1123
2668 1124
2669 1125
2670 1126
2671 1127
2672 1128
2673 1129
2674 1130
2675 1131
2676 1132
2677 1133
2678 1134
2679 1135
2680 1136
2681 1137
2682 1138
2683 1139
2684 1140
2685 1141
2686 1142
2687 1143
2688 1144
2689 1145
2690 1146
2691 1147
2692 1148
2693 1149
2694 1150
2695 1151
2696 1152
2697 1153
2698 1154
2699 1155
2700 1156
2701 1157
2702 1158
2703 1159
2704 1160
2705 1161
2706 1162
2707 1163
2708 1164
2709 1165
2710 1166
2711 1167
2712 1168
2713 1169
2714 1170
2715 1171
2716 1172
2717 1173
2718 1174
2719 1175
2720 1176
2721 1177
2722 1178
2723 1179
2724 1180
2725 1181
2726 1182
2727 1183
2728 1184
2729 1185
2730 1186
2731 1187
2732 1188
2733 1189
2734 1190
2735 1191
2736 1192
2737 1193
2738 1194
2739 1195
2740 1196
2741 1197
2742 1198
2743 1199
2744 1200
2745 1201
2746 1202
274
```

```

1620
1621    72      - transfer_asset: a valid asset suitable for handoff in transfer
1622      ↳ region.
1623    73      - collab_type: collaborative type (Type-ID).
1624    74      - OUTPUT:
1625      75      - efficient: True/False
1626    76      """
1627    77      Use the following step-by-step reasoning process to ensure the plan
1628      ↳ satisfies all constraints and results in a valid cooperative
1629      ↳ execution plan.
1630    78      """
1631    79      1. Randomly select a valid start region for robot_2:
1632      80      - Randomize among qualified candidates while maximizing path
1633      ↳ diversity.
1634      81      - Must NOT be adjacent to robot_1's start region.
1635      82      - Must be connected via the scene graph to the target object region.
1636    83      2. Determine whether collaboration is necessary, and choose the most
1637      ↳ suitable collaborative type from the following, based on robot_1
1638      ↳ and robot_2's spatial distance and task logic:
1639      84      - Type-A1: robot_2 picks up the object and delivers it to a transfer
1640      ↳ region; robot_1 takes over and completes the task.
1641      85      - Type-A2: robot_1 brings the object to a transfer region; robot_2
1642      ↳ takes over and completes the task.
1643      86      - If multiple types are valid, sample one to improve task diversity.
1644    87      3. Based on the chosen collaborative type and robot_2's region, decide
1645      ↳ whether a transfer region is required. If required:
1646      88      - It must be reachable by both robot_1 and robot_2.
1647      89      - It must **not** same as** the target object region.
1648      90      - It must contain a valid asset suitable for handoff (e.g., shelf,
1649      ↳ table, desk).
1650    91      4. Validate all pairwise constraints using tool calls:
1651      92      - You MUST call `check_collab_path_efficient_sim_graph()` with:
1652      93      - robot_2's proposed start region,
1653      94      - the selected transfer region,
1654      95      - the selected transfer asset,
1655      96      - and the chosen collaborative type.
1656      97      - If the returned `efficient` is True:
1657      98      - Must not re-call `check_collab_path_efficient_sim_graph()` .
1658      99      - you MUST remember the exact triplet of: robot_2_start_region,
1659      ↳ transfer_region, transfer_asset, collab_type to finish the
1660      ↳ plan.
1661      101     - If the returned `efficient` is False:
1662      102     - You MUST restart the planning process:
1663      103     - Re-sample a new robot_2 start region and/or a new transfer
1664      ↳ region.
1665      104     - You MUST discard the previous inefficient regions.
1666      105     - Then, you MUST re-call `check_collab_path_efficient_sim_graph()` .
1667      106     - on the new setup.
1668      107     - Repeat until a valid (efficient: True) combination is found.
1669    108      5. Output the result in the following strict JSON format:
1670    109      {
1671      110      "phase_2": {
1672      111      "Collaborative robot start region": {
1673      112      "robot_1": "RegionType_RegionID",
1674      113      "robot_2": "RegionType_RegionID"
1675      114      },
1676      115      "Collaborative type": "Type-ID",
1677      116      "Transfer region": "RegionType_RegionID",
1678      117      "Subtask instruction": {
1679      118      "robot_1": "...",
1680      119      "robot_2": "..."
1681      120      },
1682      121      "Subtask list": {
1683      122      "robot_1": [

```

```

1674
1675 121      "Move_to('..._RegionID')",
1676 122      "Grab('object_name')",
1677 123      "Move_to('..._RegionID')",
1678 124      "Release('object_name')"
1679 125  ],
1680 126  "robot_2": [
1681 127      "Move_to('..._RegionID')",
1682 128      "Grab('object_name')",
1683 129      "Move_to('..._RegionID')",
1684 130      "Release('object_name')"
1685 131  ]
1686 132  }
1687 133  }
1688 134  }
1689 135  """
1690 136 GENERAL CONSTRAINTS:
1691 137  """
1692 138  - Always respect scene graph connectivity ("link" field).
1693 139  - Use region function and assets to determine pickup and dropoff
1694 140  -> points.
1695 141  - Once a function call to `check_collab_path_efficient_sim_graph()`
1696 142  -> returns `efficient: true`, you MUST remember the exact triplet of:
1697 143  - robot_2_start_region
1698 144  - transfer_region
1699 145  - transfer_asset
1700 146  - collab_type
1701 147  - Your final output MUST use exactly the same triplet that was
1702 148  -> confirmed as efficient.
1703 149  - You MUST NOT generate new combinations without validating them
1704 150  -> again.
1705 151  - Output JSON must include "phase_2" key with the described structure.
1706 152  -> Do not explain your reasoning in the final answer.
1707 153  - Additional strict naming rules (Move_to arguments):
1708 154  -> Format must be Move_to('<entity>_<RegionID>'), **must not** use
1709 155  -> the RegionType (eg. Hallway, Bedroom, Tie) as <entity>.
1710 156  - transfer region **must not** be same as target object region.

1711 157 Here is an example of the INPUT and OUTPUT: INPUT:
1712
1713 scene graph: {"floor_1": {"region": [{"id": "Bedroom_1"}, {"id": "Bathroom_2"}, {"id": "Kitchen_4"}, {"id": "Living_room_5"}, {"id": "Hallway_6"}, {"id": "Bathroom_7"}, {"id": "Laundry_room_8"}, {"id": "Lounge_Waiting_Room_9"}, {"id": "Entryway_Foyer_10"}, {"id": "Bedroom_11"}, {"id": "Bathroom_12"}, {"id": "Bedroom_13"}], "link": [{"Bedroom_1": "Bathroom_2", "Bedroom_1": "Kitchen_4", "Kitchen_4": "Living_room_5", "Kitchen_4": "Hallway_6", "Kitchen_4": "Bathroom_7", "Kitchen_4": "Laundry_room_8", "Living_room_5": "Hallway_6", "Living_room_5": "Lounge_Waiting_Room_9", "Living_room_5": "Entryway_Foyer_10", "Hallway_6": "Laundry_room_8", "Hallway_6": "Lounge_Waiting_Room_9", "Hallway_6": "Entryway_Foyer_10", "Hallway_6": "Bedroom_11", "Hallway_6": "Bathroom_12", "Bathroom_12": "Bedroom_13"}], "item": [{"region": "Bedroom_1", "asset": ["bed", "coffee_table", "door"], "object": ["vase", "ottoman", "lamp"]}, {"region": "Bathroom_2", "asset": ["bathroom_counter", "sink", "bathroom_cabinet"], "object": ["shower_cabin", "rug", "lamp"]}, {"region": "Kitchen_4", "asset": ["dishwasher", "oven", "stove"], "object": ["flower", "flower_vase", "bowl"]}, {"region": "Living_room_5", "asset": ["shelf", "couch", "tv"], "object": ["magazine", "box", "cushion"]}, {"region": "Hallway_6", "asset": ["door"], "object": ["lamp", "vent", "rack"]}, {"region": "Bathroom_7", "asset": ["toilet", "toilet_paper", "sink"], "object": ["towel", "soap_dish", "dustbin"]}, {"region": "Laundry_room_8", "asset": ["counter", "door", "bench"], "object": []}]

```

```

1729
1730     ["basket", "coat_hanger", "lamp"]}, {"region": "
1731 Lounge_Waiting_Room_9", "asset": ["chair", "table"], "object": [""
1732 flower_vase", "rack", "flatware"]}, {"region": "Entryway_Foyer_10"
1733 ", "asset": ["door", "table"], "object": ["lamp"]}, {"region": ""
1734 Bedroom_11", "asset": ["door", "bed", "dresser"], "object": [""
1735 plant", "lamp", "box"]}, {"region": "Bathroom_12", "asset": [""
1736 bathroom_counter", "toilet", "toilet_paper"], "object": [""
1737 shower_handle", "rug", "towel_bar"]}, {"region": "Bedroom_13", ""
1738 asset": ["nightstand", "door", "bed"], "object": ["rug", "vent", ""
1739 lamp"]}}}
1740 single agent task: {"phase_1": {"Single robot start position": {""
1741 robot_1": "Bedroom_1"}, "Single robot target object position": {""
1742 robot_1": "Bedroom_13"}, "Target object": "rug", "Single robot end"
1743 position": {"robot_1": "Bathroom_2"}, "Single robot travel path": {""
1744 robot_1": ["Bedroom_1 -> Kitchen_4", "Kitchen_4 -> Living_room_5"
1745 ", "Living_room_5 -> Hallway_6", "Hallway_6 -> Bathroom_12", ""
1746 Bathroom_12 -> Bedroom_13", "Bedroom_13 -> Bathroom_12", ""
1747 Bathroom_12 -> Hallway_6", "Hallway_6 -> Living_room_5", ""
1748 Living_room_5 -> Kitchen_4", "Kitchen_4 -> Bedroom_1", "Bedroom_1 ->
1749 Bathroom_2"]}, "Task instruction": "Take the rug in bedroom to the
1750 bathroom counter in Bathroom."}}
1751
1752 OUTPUT:
1753
1754 {
1755 "phase_2": {
1756     "Collaborative robot start region": {
1757         "robot_1": "Bedroom_1",
1758         "robot_2": "Hallway_6"
1759     },
1760     "Collaborative type": "Type-A1",
1761     "Transfer region": "Kitchen_4",
1762     "Subtask instruction": {
1763         "robot_1": "Take the rug from shelf in Kitchen to the bathroom
1764             counter in Bathroom.",
1765         "robot_2": "Take the rug in bedroom to the shelf in Kitchen."
1766     },
1767     "Subtask list": {
1768         "robot_1": [
1769             "Move_to('shelf_4')",
1770             "Grab('rug')",
1771             "Move_to('bathroom_counter_2')",
1772             "Release('rug')"
1773         ],
1774         "robot_2": [
1775             "Move_to('rug_13')",
1776             "Grab('rug')",
1777             "Move_to('shelf_4')",
1778             "Release('rug')"
1779         ]
1780     }
1781 }
1782 }

```

A.11.2 SKILL FUNCTION

```
1777 1 def check_collab_path_efficient_sim_graph( 1778 2         robot_2_region, 1779 3         transfer_nodes, 1780 4         transfer_asset, 1781 5         collab_type, 1781 6         solo_cost, # 1781 7         r1_nodes,
```

```

1782
1783     target_nodes,
1784     end_nodes,
1785     region_objects,
1786     G):
1787     r2_nodes = cluster_center_node('_'+robot_2_region.split('_')[-1],
1788     ↪ nx.nodes(G), nx.get_node_attributes(G, 'position'))
1789
1790     if r2_nodes == None:
1791         return {"efficient": False}
1792
1793     print("r2_nodes:", r2_nodes)
1794
1795     if transfer_nodes:
1796         print("transfer_asset: ", transfer_asset)
1797         transfer_asset_pos = find_target_position(region_objects,
1798         ↪ '_'+transfer_nodes.split('_')[-1], '
1799         ↪ '.join(transfer_asset.split('_')))
1800         if transfer_asset_pos != None:
1801             transfer_nodes, _, _ =
1802             ↪ nearest_navpoint_to_object_vec(transfer_asset_pos,
1803             ↪ '_'+transfer_nodes.split('_')[-1], nx.nodes(G),
1804             ↪ nx.get_node_attributes(G, 'position'))
1805             insert_temp_point("transfer_"+transfer_asset, G,
1806             ↪ transfer_asset_pos, transfer_nodes)
1807             transfer_nodes = "transfer_"+transfer_asset
1808
1809     if transfer_nodes != None:
1810         print("transfer_nodes", transfer_nodes)
1811     else:
1812         return {"efficient": False}
1813     else:
1814         return {"efficient": False}
1815
1816     if collab_type == "Type-A1":
1817
1818         try:
1819             r2_to_target = nx.shortest_path_length(G,
1820             source=r2_nodes,
1821             target=target_nodes,
1822             weight='weight')
1823         except (nx.NetworkXNoPath, nx.NodeNotFound):
1824             return {"efficient": False}
1825
1826         try:
1827             target_to_transfer = nx.shortest_path_length(G,
1828             source=target_nodes,
1829             target=transfer_nodes,
1830             weight='weight')
1831         except (nx.NetworkXNoPath, nx.NodeNotFound):
1832             return {"efficient": False}
1833
1834         robot2_first_leg = r2_to_target + target_to_transfer
1835
1836         try:
1837             r1_to_transfer = nx.shortest_path_length(G,
1838             source=r1_nodes,
1839             target=transfer_nodes,
1840             weight='weight')
1841         except (nx.NetworkXNoPath, nx.NodeNotFound):
1842             return {"efficient": False}
1843
1844         parallel_leg = max(robot2_first_leg, r1_to_transfer)
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865

```

```

183666
183767     try:
183868         transfer_to_end = nx.shortest_path_length(G,
183969                         source=transfer_nodes,
184070                         target=end_nodes,
184171     except (nx.NetworkXNoPath, nx.NodeNotFound):
184272         return {"efficient": False}
184373
184474     g_parallel_cost = parallel_leg + transfer_to_end
184575     r1_parallel_cost = r1_to_transfer + transfer_to_end
184676
184777     path_infor = {
184878         "r2_to_target": r2_to_target,
184979         "target_to_transfer": target_to_transfer,
185080         "robot2_first_leg": robot2_first_leg,
185181         "r1_to_transfer": r1_to_transfer,
185282         "parallel_leg": parallel_leg,
185383         "transfer_to_end": transfer_to_end,
185484         "type": collab_type
185585     }
185686
185787     print("g_rate: ", g_parallel_cost / solo_cost)
185888     print("r1_rate: ", r1_parallel_cost / solo_cost)
185989
186090     return {
186191         "g_efficient": g_parallel_cost < solo_cost,
186292         "r1_efficient": r1_parallel_cost < solo_cost,
186393         "efficient": True if g_parallel_cost < solo_cost or
186494             ↳ r1_parallel_cost < solo_cost else False,
186595         "g_rate": g_parallel_cost / solo_cost,
186696         "r1_rate": r1_parallel_cost / solo_cost,
186797         'path_info': path_infor
186898     }
186999
1870100    elif collab_type == "Type-A2":
1871101
1872102        try:
1873103            r1_to_target = nx.shortest_path_length(G,
1874104                            source=r1_nodes,
1875105                            target=target_nodes,
1876106                            weight='weight')
1877107        except (nx.NetworkXNoPath, nx.NodeNotFound):
1878108            return {"efficient": False}
1879109
1880110        try:
1881111            target_to_transfer = nx.shortest_path_length(G,
1882112                            source=target_nodes,
1883113                            target=transfer_nodes,
1884114                            weight='weight')
1885115        except (nx.NetworkXNoPath, nx.NodeNotFound):
1886116            return {"efficient": False}
1887117
1888118        robot1_first_leg = r1_to_target + target_to_transfer
1889119
1890120        try:
1891121            r2_to_transfer = nx.shortest_path_length(G,
1892122                            source=r2_nodes,
1893123                            target=transfer_nodes,
1894124                            weight='weight')
1895125        except (nx.NetworkXNoPath, nx.NodeNotFound):
1896126            return {"efficient": False}
1897127
1898128        parallel_leg = max(robot1_first_leg, r2_to_transfer)
1899129

```

```

1890
1891     try:
1892         transfer_to_end = nx.shortest_path_length(G,
1893                                         source=transfer_nodes,
1894                                         target=end_nodes,
1895                                         weight='weight')
1896     except (nx.NetworkXNoPath, nx.NodeNotFound):
1897         return {"efficient": False}
1898
1899     g_parallel_cost = parallel_leg + transfer_to_end
1900     r1_parallel_cost = r1_to_target + target_to_transfer
1901
1902     path_infor = {
1903         "r1_to_target": r1_to_target,
1904         "target_to_transfer": target_to_transfer,
1905         "robot1_first_leg": robot1_first_leg,
1906         "r2_to_transfer": r2_to_transfer,
1907         "parallel_leg": parallel_leg,
1908         "transfer_to_end": transfer_to_end,
1909         "type": collab_type
1910     }
1911
1912     print("g_rate: ", g_parallel_cost / solo_cost)
1913     print("r1_rate: ", r1_parallel_cost / solo_cost)
1914
1915     return {
1916         "g_efficient": g_parallel_cost < solo_cost,
1917         "r1_efficient": r1_parallel_cost < solo_cost,
1918         "efficient": True if g_parallel_cost < solo_cost or
1919             ↪ r1_parallel_cost < solo_cost else False,
1920         "g_rate": g_parallel_cost / solo_cost,
1921         "r1_rate": r1_parallel_cost / solo_cost,
1922         'path_info': path_infor
1923     }
1924
1925     else:
1926         return {
1927             "efficient": False
1928         }
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

```

1944 A.12 PROMPTS FOR GRAPH CONTEXTUAL TYPING
19451946 Room Type Reasoning
19471948 **System:**

1949 You are an AI assistant that specializes in spatial reasoning and semantic scene understanding. Your task
1950 is to infer the most likely type of a room (e.g., Bedroom, Bathroom, Kitchen, Laundry room, Living room,
1951 Hallway, Tie, Balcony, Terrace, etc.) based on the provided contextual clues.

1952 **Rules:** There are two part of the input: neighbors and objects. The neighbors part includes information
1953 about the number and types of adjacent rooms. This gives spatial and functional context, which helps
1954 narrow down the likely use of the unknown room. The objects part lists the items found in the unknown
1955 room. These object types and frequencies are strong indicators of the room's function.

1956 1 When inferring the type of an unknown room, your reasoning must be
1957 → guided by both the semantic distribution of objects and the
1958 → spatial context of neighboring rooms.
1959 " "
1960 2 - Always prioritize distinctive, functionally indicative objects over
1961 → generic decorative ones.
1962 3 - Use neighboring room types to constrain plausible options (e.g., a
1963 → room between two Bathrooms is unlikely to be a Kitchen).
1964 4 - Avoid inferring ambiguous multifunctional rooms unless object
1965 → diversity strongly supports it.
1966 5 - Room type must be one from the predefined category list (e.g.,
1967 → Bedroom, Bathroom, Kitchen, etc.).
1968 6 - In cases where the object list is sparse, weigh neighbor consistency
1969 → more heavily.
1970 " "
1971 9 Below is the full list of supported room types, each with a brief
1972 → description of its primary function.
1973 " "
1974 10 - Bedroom | used for sleeping and personal rest.
1975 11 - Bathroom | supports hygiene tasks like bathing and toileting.
1976 12 - Kitchen | designed for cooking and food preparation.
1977 13 - Dining Room | a place for eating meals, often next to the kitchen.
1978 14 - Living Room | used for leisure, social interaction, or
1979 15 → entertainment.
1980 16 - Study / Office | a workspace for reading, writing, or computer use.
1981 17 - Laundry Room | contains appliances and tools for washing clothes.
1982 18 - Closet / Storage Room | used to store clothes, tools, or household
1983 19 → items.
1984 20 - Hallway / Corridor | connects other rooms; mainly transitional.
1985 21 - Garage | for parking vehicles or storing tools and equipment.
1986 22 - Kids Room / Nursery | a bedroom tailored for children, often with
1987 23 → toys.
1988 24 - Balcony / Terrace | a semi-outdoor area for air, light, or drying.
1989 25 - Media Room / Home Theater | equipped for movies or audio-visual
1990 26 → activities.
1991 27 - Gym / Fitness Room | contains equipment for exercise and physical
1992 28 → training.
1993 29 - Library | a quiet space for reading or storing books.
1994 30 - Meeting Room | a formal area for group discussions or presentations.
1995 31 - Lounge / Waiting Room | a rest area in public or semi-public
1996 32 → buildings.
1997 33 - Pantry / Bar | a compact space for storing or serving drinks/snacks.
1998 34 - Dressing Room | dedicated to changing clothes or grooming.
1999 35 - Entryway / Foyer | the front entrance space where people enter the
1999 36 → home
" "
Output format constraints:
" "
- Only output the final predicted room type.
- The output must be a single Python dictionary with the format:
{ "Unknown room type": "<Room Name>"}

```

1998
1999 37 - Do NOT include any reasoning, explanation, or analysis in the
2000    ↳ output.
2001 38 - Do NOT output multiple room types or probabilities|only the most
2002    ↳ likely one.
2003 39 - Do NOT include any commentary, bullet points, or markdown
2004    ↳ formatting.
2004 40 """
2005 41 Here is an example of the INPUT and OUTPUT:
2006 42 INPUT:
2006 43 """
2007 44 neighbors: "There are 2 neighboring rooms, belonging to 1 type,
2008    ↳ including 2 Kitchens."
2009 45 objects: "The unknown room contains 77 objects, belonging to 30 types.
2010    ↳ The top 5 most frequent items are: 22 photos, 11 chairs, 3 plants,
2011    ↳ 3 vases, and 3 shelfs."
2011 46 """
2012 47 OUTPUT:
2013 48 """
2014 49 {"Unknown room type": "Living room"}
2015 50 """

```

A.13 ABLATION FOR PROMPTS

A.13.1 DIRECTLY TWO-AGENT TASK GENERATION

V1 and V2 demonstrate the following shortcoming ...



Figure 17: Common failure case of directly generating a two-agent task.

In fact, the initial versions of CoNavBench were exactly based on directly generating two-agent tasks, as you suggested, and our current “single-agent task then split” pipeline is the result of several iterations driven by empirical observations. Concretely, we implemented and tested three major versions of the data-generation pipeline on a fixed test set of HM3D environments(00087-YY8rqV6L6rf,00299-bdp1XNEdvmW,00323-yHLr6bvWsVm,00324-DoSbs0o4EAg,00444-sX9xad6ULKc,00612-GsQBY83r3hb) to allow controlled comparisons under identical prompts. However, after extensive testing, we found that both V1 and V2 suffered from very low-quality multi-robot collaboration, mainly due to the difficulty of long-horizon spatial reasoning with two agents at once. Typical failure modes included: I) Little or no true parallelism: the LLM often produced plans where one robot did almost all the work while the other remained idle, so executing the plan with a single robot sequentially was actually more

2052 efficient, as shown in Figure 17. **II) Inconsistent or invalid coordination:** mismatches between the
 2053 transfer region and the subtask descriptions, missing handover steps, or paths that did not align with
 2054 the scene layout. Hence, motivated by these observations, we adopt the two-stage NavCraft design.
 2055 We have added this discussion in the Supplements Section for more details.
 2056

2057

2058 Directly Two-agent Task Rule Prompt Version 1

2059

```

2060 1 Important rules regarding the scene graph:
2061 2 """
2062 3 - You must leverage the scene graph topology to design meaningful
2063 4   ↳ multi-robot cooperation.
2064 5 - The connectivity between regions (provided in the "link" field of
2065 6   ↳ the scene graph) must be used to ensure:
2066 7   - Robots do not start in adjacent regions.
2067 8   - Task handoffs (e.g., object relay between robots) happen in
2068 9   ↳ regions that are connected.
2069 10 - The selection of regions and object transfer paths should reflect
2070 11   ↳ realistic spatial planning based on the graph.
2071 12 - Encourage designs where the first robot delivers an object to an
2072 13   ↳ intermediate node (transfer zone), and the second robot continues
2073 14   ↳ from there. This creates natural cooperation patterns.
2074 15 """
2075 16 There are something you need to pay attention to:
2076 17 """
2077 18 - A robot should not start in the same region where it needs to pick
2078 19   ↳ up or drop off objects.
2079 20 - Each robot's contribution should reduce the overall task cost:
2080 21   - Avoid assigning tasks that could be completed more efficiently
2081 22   ↳ by a single robot.
2082 23   - Only use multi-robot handoff when it significantly reduces
2083 24   ↳ travel distance or enables parallelism.
2084 25 - The multi-robot plan you generate should not be less efficient than
2085 26   ↳ a single-robot plan for the same task. Cooperation should lead to
2086 27   ↳ either reduced execution time or more balanced workload across
2087 28   ↳ agents.
2088 29 - The objects involved must be portable and must appear in the input
2089 30   ↳ scene.
2090 31 - The task must involve only 1 to 2 different regions.
2091 32 - The region IDs (e.g., "Kitchen_1", "Bedroom_3") do not imply spatial
2092 33   ↳ proximity or connectivity. Only the "link" field in the scene
2093 34   ↳ graph provides valid region-to-region connections. Do not assume
2094 35   ↳ regions with similar names or IDs are connected.
2095 36 - The full task must contain 4 to 6 subtasks total (across all
2096 37   ↳ robots).
2097 38 - The task you generate should be similar to instructions like "Take
2098 39   ↳ an object in one region to a certain place in one region."
2099 40 - Subtasks must follow logical ordering: a robot must Move_to before
2100 41   ↳ Grab or Release; it cannot Release without having grabbed the
2101 42   ↳ object first.
2102 43 - The region mentioned in Move_to() should match the region mentioned
2103 44   ↳ in the high-level instruction.
2104 45 - The task must reflect multi-robot cooperation, such as transporting
2105 46   ↳ an object to a place where another robot picks it up and
   ↳ continues.
   - Do not use low-level action terms like "grab", "release", or
   ↳ "move_to" in "Task instruction" or "Subtask instruction".
"""
2101 27 Your output should be a Python dictionary with the following keys:
2102 28 """
2103 29 - "Robots position": A dictionary assigning each robot to a starting
2104 30   ↳ region. Each robot must start in a different region, and the
2105 31   ↳ regions must be at least one hop apart in the scene graph.

```

```

2106
2107 30 - "Task instruction": A conversational high-level instruction
2108    ↳ describing the overall collaborative task.
2109 31 - "Transfer position": A list of intermediate regions (e.g., region
2110    ↳ IDs) that serve as **handover zones** between robots. These
2111    ↳ positions should be selected from the scene graph and must be
2112    ↳ reachable by both the sender and the receiver robot. Use them to
2113    ↳ support collaborative efficiency: for example, Robot A may carry
2114    ↳ an object from Region X to a transfer position, where Robot B
2115    ↳ picks it up and delivers it to the final destination. When
2116    ↳ selecting a transfer position, prefer regions that minimize the
2117    ↳ total travel distance between participating robots.
2118 32 - "Subtask instruction": A dictionary giving each robot a
2119    ↳ natural-language description of its individual role in the task.
2120    ↳ Avoid technical terms like "grab" or "release".
2121 33 - "Subtask list": A dictionary mapping each robot to a list of
2122    ↳ low-level subtasks. These subtasks should be composed of the
2123    ↳ following actions:
2124    - Move_to("object_region_id"): Walk to an object or location in a
2125      ↳ region. The "object_region_id" must combine the object name and
2126      ↳ its region ID from the input scene (e.g., "lamp_1").
2127    - Grab("object"): Pick up the object. The robot must first move to
2128      ↳ the object's location. The object must exist in the "objects"
2129      ↳ list of the input scene.
2130    - Release("object"): Place the object in the target asset or
2131      ↳ location. The robot must first move to the target. The object
2132      ↳ must exist in the input scene.
2133 37 """
2134 38 Make sure the task instruction conversational enough, and the task
2135    ↳ should reasonable.
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159

```

Directly Two-agent Task Rule Prompt Version 2

```

2135 1 Important rules regarding scene and task planning:
2136 2 """
2137 3 - Use the region connectivity graph ("link") to determine all movement
2138    ↳ and transfer feasibility.
2139 4 - Robots must start in **different**, **non-adjacent** regions.
2140 5 - A robot cannot start in the same region where it will pick up or
2141    ↳ drop off an object.
2142 6 - Task handoff must happen in a region reachable from both the sender
2143    ↳ and receiver.
2144 7 - A robot must Move_to a region before performing Grab or Release.
2145 8 - A robot must Grab an object before it can Release it.
2146 9 - The region specified in Move_to() must match the one implied in the
2147    ↳ instruction (no teleportation).
2148 10 - Subtasks must follow this logical order: Move_to → Grab → Move_to →
2149    ↳ Release.
2150 11 - The objects involved must:
2151    - Be listed in the input scene's "object" field (i.e., exist and be
2152      ↳ portable).
2153    - Be manipulated **only**, not assets.
2154 13 - Subtasks must not violate logical flow or act on unavailable
2155    ↳ objects/assets.
2156 15 - The overall plan must involve **1 to 2 regions total** (e.g., task
2157    ↳ origin, destination, or transfer area).
2158 16 - Tasks must be split into **4 to 6 total low-level subtasks**, across
2159    ↳ all robots.
2160 - Task planning should reflect genuine cooperation:
2161    - Prefer parallelism or reduced path cost through collaboration.
2162    - Avoid plans where a single robot could complete the task more
2163      ↳ efficiently.

```

```

2160
2161 20      - Multi-robot plans **must not be less efficient** than single-robot
2162      ↳ alternatives.
2163 21      - The task must exhibit **explicit collaboration**, such as robot_1
2164      ↳ transporting an object to an intermediate location, and robot_2
2165      ↳ completing delivery.
2166 22      - Avoid assuming proximity based on region name:
2167      - Region IDs (e.g., "Kitchen_1", "Kitchen_2") **do not imply spatial
2168      ↳ adjacency**.
2169 23      - Use only "link" data to determine connectivity.
2170 24      - Task instruction and subtask instructions should:
2171      - Be natural-language (e.g., "Take the object from A to B").
2172      - **Avoid technical terms** like "grab", "release", or "move_to".
2173 25      """
2174 26      Region Symbol Definitions (used in output and reasoning):
2175 27      """
2176 28      - Region_A: Initial position of robot_1
2177 29      - Region_B: Location of the portable object
2178 30      - Region_X: Handoff region (object relay from robot_1 to robot_2)
2179 31      - Region_Y: Initial position of robot_2
2180 32      - Region_C: Final destination for object delivery
2181 33      """
2182 34      Spatial Constraints:
2183 35      """
2184 36      - Region_A and Region_Y must NOT be adjacent.
2185 37      - Region X and Region B must NOT be adjacent.
2186 38      - Region X and Region C must NOT be adjacent.
2187 39      - Among Region_A, Region_X, Region_Y, and Region_C, Region_B must be
2188 40      ↳ the closest to Region_A based on the region connectivity graph
2189 41      ↳ (i.e., shortest path length from Region_A).
2190 42      - Among Region_A, Region_X, Region_Y, and Region_B, Region_C must be
2191 43      ↳ the closest to Region_Y based on the region connectivity graph
2192 44      ↳ (i.e., shortest path length from Region_Y).
2193 45      """
2194 46      Output Format:
2195 47      """
2196 48      {
2197 49      "Robots start position": {"robot_1": "Region_A", "robot_2":
2198 50      ↳ "Region_Y"},
2199 51      "Transfer position": "Region_X",
2200 52      "Robot travel path": {
2201 53      "robot_1": ["Region_A -> Region_B", "Region_B -> Region_X"],
2202 54      "robot_2": ["Region_Y -> Region_X", "Region_X -> Region_C"]
2203 55      },
2204 56      "Task instruction": "Take the [object_B] from Region_B to the
2205 57      ↳ [asset_C] in Region_C.",
2206 58      "Subtask instruction": {
2207 59      "robot_1": "Take [object_B] in Region_B to the [asset_X] in
2208 60      ↳ Region_X.",
2209 61      "robot_2": "Take [object_B] from [asset_X] in Region_X to the
2210 62      ↳ [asset_C] in Region_C."
2211 63      },
2212 64      "Subtask list": {
2213 65      "robot_1": ["Move_to('object_B')", "Grab('object')",
2214 66      ↳ "Move_to('asset_X')", "Release('object')"],
2215 67      "robot_2": ["Move_to('asset_X')", "Grab('object')",
2216 68      ↳ "Move_to('asset_C')", "Release('object')"]
2217 69      }
2218 70      }
2219 71      """
2220 72      Output field explanation:
2221 73      """
2222 74      - "Robots start position": Dict mapping each robot to its starting
2223 75      ↳ region (must follow spatial constraints).

```

```

2214
2215 68 - "Transfer position": Region ID (or list of region IDs) where object
2216 69   ↳ is passed from robot_1 to robot_2.
2217 69 - "Robot travel path": Dict showing robot movement as ordered
2218 70   ↳ region-to-region transitions.
2219 70 - "Task instruction": Natural description of full multi-robot delivery
2220 70   ↳ task.
2221 71 - "Subtask instruction": A dictionary giving each robot a
2222 71   ↳ natural-language description of its individual role in the task.
2223 71   ↳ Avoid technical terms like "grab" or "release".
2224 72 - "Subtask list": Dict of robot action sequences using these
2225 72   ↳ primitives:
2226 73   - Move_to("object_region_id")
2227 73   - Grab("object")
2228 73   - Release("object")
2229 76 """
2230 77 Make sure the task instruction conversational enough, and the task
2231 77   ↳ should be reasonable.
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267

```