

Commonsense-Grounded Path Planning from Abstract Instructions

Masafumi Endo¹, Kohei Honda^{1,2}, Ryo Yonetani¹

Abstract—We present *commonsense ranked search (CoRS)*, a novel path planner that turns an abstract instruction into a route that follows commonsense. While existing methods respect the considerations written down in advance, a robot working among people must follow those left unstated too, as with a wet floor that a worker avoids without being told. CoRS leverages large language models (LLMs) and vision-language models (VLMs) as commonsense knowledge to reason about these latent considerations in its planning. Given an abstract instruction (e.g., “move carefully”) and visual observations of each region in the environment, CoRS derives a consideration for each region, as in “this wet floor is slippery and worth a detour.” It then compares the considerations between regions to see which of the two the robot should avoid more, as in “the crowd is worse than the wet floor.” These judgments sort the regions into a commonsense ranking, whose costs drive a conventional search that always returns a valid route. We build a benchmark for planning under latent considerations, with three environments, 1350 problems, and five instructions at three levels of abstraction. Experiments show that CoRS discovers the unstated considerations and goes around the ones worth a detour while crossing the rest, a behavior that recent LLM-based planners do not achieve.

I. INTRODUCTION

Society expects us to act with commonsense. Imagine a supermarket worker is asked to “carry this basket to the backroom carefully.” Tactile paving lines the floor, customers queue at the checkout, and an aisle is wet after cleaning. The worker hears nothing more than *carefully*, yet adjusts the route so that each of them gets the right amount of care. Commonsense carries us from abstract instructions to concrete decisions, on which everyday life runs smoothly.

Our goal is to make mobile robots move among people with such commonsense. These robots have to turn the given instruction into a route (Fig. 1). Each situation on its way puts a *consideration* on the robot: risk, constraint, or preference. Existing work has incorporated such considerations into planning as costs translated from instructions [1]–[3], regions on the map that language can refer to [4], [5], or rules with priorities set in advance [6]–[9]. However, these methods assume every consideration to be written down explicitly, rather than latent behind an abstract instruction.

We thus propose *commonsense-grounded path planning*, a novel planning problem for a robot that follows an abstract instruction. Given the instruction and a collection of

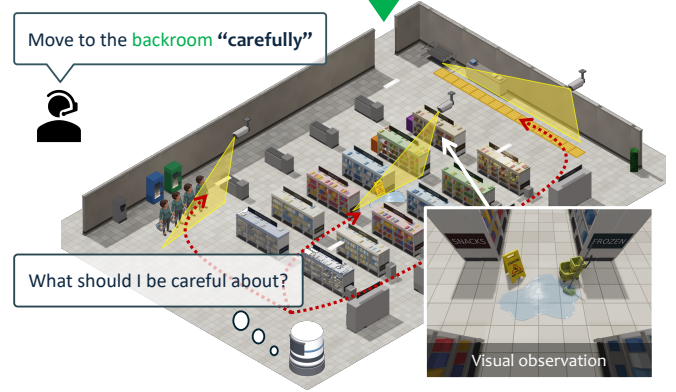


Fig. 1. **Motivating scenario: Planning by commonsense.** In our simulator, a robot is asked to move *carefully* in a supermarket and sees each region through a surveillance camera. The instruction does not say which regions to avoid. No route avoids all of them, so it must decide which to cross.

visual observations that cover multiple regions across the environment, the robot must find the right route without explicit considerations to respect. The challenge is that the considerations change with the instruction, the task, and the remaining routes together. Suppose the robot finds a wet aisle on its route. Usually it can cross, but with a cup of coffee to deliver it should take a detour. When no detour exists, it should cross rather than abandon the task. Large language models (LLMs) and vision-language models (VLMs) can judge such situations at planning time [1], [10]. Yet the model may write a route that the map does not have [11]–[13], or select among the candidates a classical planner prepared from geometry alone [10], [14].

As a solution to this problem, we propose **commonsense ranked search (CoRS)**. Given an abstract instruction and visual observations, it leverages LLMs and VLMs to infer the latent considerations and turn them into the costs of search-based planning. It first describes the observation of each region as text through a VLM. It then puts two questions to an LLM: whether the region is worth a detour and which of two regions the robot should avoid more. The answers sort the regions against one another and against the length of the route into a *commonsense ranking*. Its costs drive a classical search and keep every route valid. The same region is thus avoided in one situation and crossed in another, as the instruction, the task, and the remaining routes change.

We evaluate CoRS in simulated everyday environments where service robots work, such as supermarkets and hospitals. We build a *commonsense planning benchmark*, a simulator and a problem set for evaluating LLM-based planners under latent considerations. We design the situations our-

This work was supported by JSPS KAKENHI Grant Number 26K02991. This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

¹CyberAgent Inc., Tokyo, Japan ²Nagoya University, Aichi, Japan
{endo.masafumi, honda.kohei, yonetani.ryo}@cyberagent.co.jp

selves and record the right behavior for each, which existing environments do not provide. Our benchmark covers three environments and 1350 problems, where five instructions at three levels of abstraction vary how much the robot works out for itself. Experiments show that CoRS finds the unstated considerations and goes around the ones worth a detour with far fewer needless detours than LLM-based planners.

Our contributions are as follows:

- **Commonsense-grounded path planning.** A new planning problem where no consideration is stated, unlike the settings that give the rules to follow in advance.
- **Commonsense ranked search.** A planner that brings an LLM into the cost of the search to weigh the regions relative to one another rather than in isolation.
- **Commonsense planning benchmark.** A simulator and a problem set that record the right behavior under latent considerations, which no existing environment provides. Fig. 1 shows one of its environments.

We will release the benchmark with its simulator and baselines upon publication.

II. RELATED WORK

We review commonsense in robot navigation, instruction-guided costs, and foundation models as planners.

a) Commonsense in robot navigation: Robot navigation among people involves not only geometric obstacles but also the considerations that hold in a place. Classical work enumerates them by hand, along with the priorities that decide which to break first [6]–[9]. Sathiamoorthy *et al.* classify the scene with a VLM, but still write the behavior for each scene by hand [15]. RoboGuard grounds the rules written down on the scene with an LLM [16]. Yet each rule is checked as a binary condition, so the robot cannot weigh how much it breaks one rule against another. Recent work instead learns them from how people already move. Choi *et al.* train a navigation policy on hours of human driving [17]. Chen *et al.* learn from expert trajectories as well, but add a reward that scores social compliance [18]. However, a robot trained this way cannot find the unstated considerations, and a recent benchmark reports far lower compliance with them than with the stated goals [19].

b) Instruction-guided cost: When the instruction states the rules to follow, a planner can encode them as the cost on the map and search for a suitable route. One approach turns the instruction into a cost, but the map holds geometry alone [3]. Other work derives the required behavior from the instruction, asks a VLM to check each place against it, and raises the cost where the place fails [1], [2]. A score from the model can also be the cost, as LFG scores each frontier for object-goal navigation [20] and Song *et al.* score the social entities around the robot for a local planner [21]. Sugino *et al.* bring it to the instruction, where a VLM scores each observation image on a graph and the score scales the edge costs [22]. While these methods cover a range of instructions, they judge each place on its own, so a place carries the same cost in any situation.

c) Foundation models as planners: Foundation models carry commonsense about the behavior that suits a place, and recent work asks them to plan the route with it. MapGPT and the NavGPT family write the places and their connections into the prompt, from which the model moves among the neighbors at each step [11]–[13]. Yet even in a small maze, an LLM writes an invalid route due to misreading the spatial relations, such as one through an obstacle or into a dead end [23]. Another approach therefore restricts the model to selecting one of the candidate routes from a classical planner [10], [14]. The candidates come from geometry alone, on the explicit assumption that the right route for the situation is among them [10].

In summary, existing work either converts the considerations written down into a fixed cost per place, or leaves the route to a foundation model outside the search. We instead integrate the model into the search, where it infers the latent considerations and ranks the regions against one another.

III. PRELIMINARIES

We describe standard forms of path planning on a topological map and the costs we build on, and then state the problem we address.

A. Path Planning on Topological Maps

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected topological map of the environment, where the vertices $\mathcal{V}$ are places and each edge $(u, v) \in \mathcal{E}$ connects places that the robot can move between, with a length $d(u, v) > 0$. A set $\mathcal{R}$ of regions represents the situations on the floor, such as a wet aisle or a queue. Each region $r \in \mathcal{R}$ covers a part of the map and has one or more observation images o_r that show the situation from above. A route $P = (v_0, \dots, v_T)$ is a sequence of places from $v_0 = v_s$ to $v_T = v_g$ with $(v_{t-1}, v_t) \in \mathcal{E}$ for all t , where v_s and v_g are a start and a goal. With an arbitrary nonnegative cost $w(u, v)$ on every edge, the cost of a route is the sum of the edge lengths and the costs of the edges along it,

$$J(P) = \sum_{t=1}^T [d(v_{t-1}, v_t) + w(v_{t-1}, v_t)]. \quad (1)$$

A classical graph search returns the route that minimizes $J(P)$, so w is the only factor we can control.

B. Consideration Levels as Costs

Each region r carries a set C_r of considerations. A consideration is a demand that the situation there puts on the robot, such as the risk of a wet floor, the constraint of a staff-only area, or the preference for quiet near a resting patient. When every route to the goal passes through a region with a consideration, the robot has to choose which one to cross. Each region with a consideration takes a level, given by a function $k : \{r \mid C_r \neq \emptyset\} \rightarrow \{1, \dots, K\}$. Let $e_r(u, v) \geq 0$ be the length of the edge (u, v) inside the region r . The levels then replace the scalar cost in (1) with a vector

$c(P) \in \mathbb{R}^{K+1}$, one component per level and the last for the length,

$$c_j(P) = \sum_{t=1}^T \sum_{r: k(r)=j} e_r(v_{t-1}, v_t) \quad \text{for } j = 1, \dots, K,$$

$$c_{K+1}(P) = \sum_{t=1}^T d(v_{t-1}, v_t). \quad (2)$$

Routes compare on $c(P)$ component by component from the first. A region of level j thus costs more than any length inside regions of levels $j' > j$ together with any route length. When every route crosses a region, the search returns the one that crosses the least at the highest level it cannot avoid. Such costs appear in planning for autonomous driving under traffic rules, where the levels are the priorities among the rules, known as minimum-violation planning and rulebooks [7]–[9]. Those methods count a violation under its rule, but keep both C_r and k fixed in advance.

C. Problem Setup

We now formalize *commonsense-grounded path planning*, where a robot finds a route on G under an instruction ℓ given in natural language. A problem instance is a tuple $(G, \mathcal{R}, \{o_r\}, v_s, v_g, \ell)$, on which we make the following three assumptions.

- 1) The instruction comes at various levels of abstraction a person would use, from an explicit rule such as “keep off the tactile paving” to a single word such as “carefully.” It may also state the task the robot is on, such as carrying glassware.
- 2) The situation in a region is known only through its observation images o_r . This holds in indoor environments such as supermarkets, where surveillance cameras already cover the floor.
- 3) Neither C_r nor the level k is given. They follow from the situation in each region, together with the instruction, the task, and the routes that remain on the map.

We call *commonsense* the judgment that derives C_r and k from the situation when the instruction leaves them unstated, where a demand on people weighs more than one on the robot. The solution is a route P on G that follows it. The standard form cannot produce it, since it takes C_r and k as given. We instead derive both at planning time, as described next.

IV. COMMONSENSE RANKED SEARCH

We present *commonsense ranked search* (CoRS), a path planner that derives C_r and k from the instruction and the observations. CoRS judges whether each region is worth a detour and which of two regions the robot should avoid more. The answers then form a *commonsense ranking*, a single order over the regions and the length of the route. A search on the costs from this ranking returns the route. Algorithm 1 summarizes the pipeline.

Instruction 1

“You carry a full tray.”

B > **A** > len > **C**

Instruction 2

“You make noise while working.”

A > **B** > len > **C**

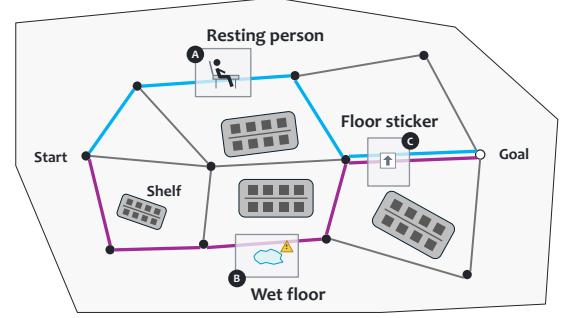


Fig. 2. Example problem instance solved by CoRS. The same regions get a different ranking and a different route under each instruction.

A. Grounding the Instruction on Each Region

CoRS first reads each region on its own (Algorithm 1, L2–L3). A VLM writes a text t_r from the observation images o_r of every region. The text is a neutral description of the observation, such as a queue at the checkout or a cart left in the aisle, for the LLM to judge. An LLM then reads ℓ and t_r and answers whether the region is worth a detour,

$$a_r = \text{WORTHDETOUR}(\ell, t_r) \in \{\text{detour}, \text{cross}, \text{none}\}, \quad (3)$$

where *none* marks a region the text gives no basis to judge. The LLM writes the reason before each answer, which states the consideration the situation puts on the robot and thus derives C_r . Under “you carry a full tray” in Fig. 2, the LLM answers *detour* for the wet floor (“it is slippery”) and *cross* for the floor sticker (“it is safe to step on”). This step decides whether a region matters, while the comparison below decides how much.

B. Comparing Two Regions

CoRS also compares the regions with one another. For a pair r, r' , an LLM reads ℓ and the two texts and answers which of the two the robot should avoid more,

$$b_{r,r'} = \text{COMPARE}(\ell, t_r, t_{r'}) \in \{r, r', \text{tie}\}, \quad (4)$$

where *tie* covers an answer that the two are alike and one that the texts give no basis to separate them. In Fig. 2, the LLM keeps the robot off the resting person under “you make noise while working” (“the noise reaches the person”). It keeps the robot off the wet floor instead under “you carry a full tray” (“a spill would be worse than the noise”). An absolute score instead tends toward caution and leaves little difference among the regions. An LLM also judges texts more reliably in pairs than by a score alone [24], [25].

Algorithm 1 Commonsense Ranked Search**Input:** problem instance $(G, \mathcal{R}, \{o_r\}, v_s, v_g, \ell)$ **Output:** route P from v_s to v_g

```

1: for  $r \in \mathcal{R}$  do
2:    $t_r \leftarrow \text{DESCRIBE}(o_r)$   $\triangleright$  VLM
3:    $a_r \leftarrow \text{WORTHDETOUR}(\ell, t_r)$   $\triangleright$  LLM
4: end for
5:  $\mathcal{L} \leftarrow \text{SORT}(\mathcal{R} \cup \{\text{len}\}, \prec)$   $\triangleright \prec$  from (5)
6:  $k \leftarrow$  level of each element in  $\mathcal{L}$ 
7:  $P \leftarrow \text{SEARCH}(G, v_s, v_g, c)$   $\triangleright c$  from  $k$  by (6)
8: return  $P$ 

```

C. Sorting into a Commonsense Ranking

The two answers above define a comparator over the regions and the length of the route, written len ,

$$x \prec y = \begin{cases} b_{x,y} & x, y \in \mathcal{R}, \\ a_x & x \in \mathcal{R}, y = \text{len}, \end{cases} \quad (5)$$

where a region goes above len when a_x is `detour` or `none`, and below it when a_x is `cross`. A region with no basis thus stays above len , as the standard form of Section III-B places every consideration. Sorting $\mathcal{R} \cup \{\text{len}\}$ under $\prec$ then returns a commonsense ranking, an order over the regions and the length with ties (L5). In Fig. 2, the floor sticker stays below len under both instructions, while the resting person and the wet floor swap above it. The sort queries the LLM only for the pairs it needs, rather than for all $|\mathcal{R}|^2$ of them.

Let the ranking have levels $1, \dots, K$, with len alone at level k_{len} and k now defined on every region (L6). Then (2) generalizes to a vector $c(P) \in \mathbb{R}^K$, one component per level and the length at component k_{len} ,

$$c_j(P) = \sum_{t=1}^T \sum_{r: k(r)=j} e_r(v_{t-1}, v_t) \quad \text{for } j \neq k_{\text{len}}, \quad (6)$$

$$c_{k_{\text{len}}}(P) = \sum_{t=1}^T d(v_{t-1}, v_t).$$

A region above len costs more than any detour around it, so the robot goes around it whenever a route exists. A region below len matters only between routes of equal length, so the robot passes it whenever the route through it is shorter.

D. Searching on the Ranking

A search-based planner on $c(P)$ returns the route (L7, Fig. 2). It compares the vector costs component by component from the first, so a lower cost at a higher level decides the route. The search visits the map itself, so the route it returns exists on G and reaches v_g whenever one does.

V. COMMONSENSE PLANNING BENCHMARK

A benchmark for commonsense planning needs three things: situations that need commonsense, problems where the right route changes, and a right route on record for each. Our *commonsense planning benchmark* prepares such situations, problems, and their right routes, with the instructions at different levels of abstraction.

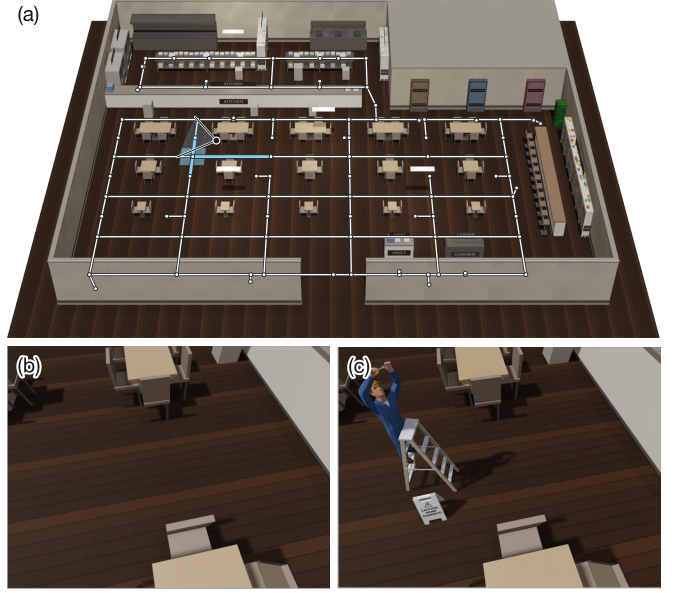


Fig. 3. Benchmark construction in the restaurant environment. (a) A region on the floor, the graph, and the camera that sees it. (b) The original view from that camera. (c) The same view with the situation drawn in.

A. Environments

Existing indoor environments are scans of real spaces or layouts built for household tasks [26]–[28], so they do not pose situations that need commonsense. Benchmarks for social navigation exist, but test the robot only against moving people [29] or the rules written down [30]. We therefore build a 2.5D simulator, where we place such situations at the places of our choice. From one floor plan each, we generate three environments: a supermarket, a restaurant, and a hospital, each with its own considerations for a service robot. They differ in the shape of the free space, from the parallel aisles of the supermarket to the lattice of tables in the restaurant, and so in the routes around a situation.

Fig. 3 shows the restaurant environment with an example situation and its observation image. The vertices of the map lie at the junctions of the aisles and in front of the landmarks, with the edges along the aisles between them (Fig. 3a). This gives a graph of about 50 to 95 places. We place the situations as regions on the floor, each a rectangle across an aisle, and an arrangement holds one to three of them. Each place has a camera that looks down at the floor, so a region appears in the images from the places around it. Writing each situation as text gives us any case we need, including those that call for complex commonsense. We use Gemini 3.1 Flash Image to draw them into the images (Fig. 3b, c).

B. Problem Types

The right route changes with the situation, the instruction, the task, and the remaining routes, so each problem type below varies one of these four factors.

- **Unstated care.** The situation changes from one worth a detour to one worth none, so the robot must choose a detour or a crossing from the situation alone.

- **Situational urgency.** The instruction changes from no hurry to a delivery due in minutes.
- **Robot duty.** The duty changes from cleaning the floor with noise to carrying items quietly, so the robot avoids a different region in each problem.
- **Available detour.** The three conditions differ in whether a route around the situation exists, and whether to take it when it does.

Every problem puts a situation on the shortest route and makes a detour cost more than a little, so the right route differs from the shortest one. A problem instance fixes an arrangement of the situations, a pair of endpoints, and an instruction. Each environment carries nine arrangements and ten pairs of endpoints in each. Available detour then holds 450 problems across the three environments, and each of the other types holds 300.

C. Levels of Abstraction

A planner also has to supply what the instruction leaves out, so the instructions come on three levels that differ in how much they state. The **coarse** level states the task with its urgency or duty, but nothing about how to move. The **base** level adds a single word such as “safely,” in three phrasings of the instruction. The **detailed** level names the behavior for each situation, along with its object where the behavior has one, as in “keep off the tactile paving.” The coarse and base levels thus leave the regions to avoid for the robot to judge, while the detailed level names them and sets a reference. Across the three environments, the evaluation covers 1350 problems that every planner solves.

D. Ground Truth and Metrics

A metric needs one right route for every problem. For every arrangement, we therefore rank the regions and define which of them the robot should avoid, and we derive the reference route from that ranking. The metrics below compare the returned route with this reference.

- **Behavior match.** The fraction of problems in which the route agrees with the reference on all three counts: which regions to avoid, which to cross, and whether to detour.
- **Critical violation.** Among the problems with a critical region, the fraction in which the route crosses it. A critical region is one the robot must not pass under any instruction, such as a worker on a ladder or an infant on the floor.
- **Over-conservatism.** Among the problems where no detour is right, the fraction in which the route takes a detour.

Per environment, behavior match covers 450 problems, critical violation 200, and over-conservatism 150.

VI. EXPERIMENTS

We designed our experiments on the benchmark in Section V to answer the following questions:

Q1. Can a planner derive the unstated considerations?

- Q2. Does a planner return a different route under the same situation when the instruction, the task, or the remaining routes change?
- Q3. Does turning the judgment into costs on the map return the right route more often than writing or selecting one?
- Q4. Does ranking the regions against one another tell them apart better than scoring each on its own?

A. Baselines

We compare CoRS with three planners that differ in how they employ LLMs and VLMs in planning.

- **Direct planning** [11]–[13]. The model writes the route from the texts and an adjacency list of the map with neutral node names. The route may be absent from the map or miss the goal, in which case the planner falls back to the shortest path.
- **Candidate selection** [10], [14]. The model selects one of m candidate routes that include the shortest one and differ from one another in shape. The candidates come from geometry alone, so the right route may not be among them.
- **Absolute scoring** [22]. The model scores each region on its own and scales the length inside the region by the score. No comparison enters the score, so the same region carries the same cost in any situation.

We also compare with **Shortest path**, which searches on the length alone, and **Uniform caution**, which places every region above the length without reading the texts. **Oracle ranking** takes the ranking we define in Section V-D and keeps every region above the length. It thus shows the gain from the right ranking alone, without the judgment against the length.

B. Implementation Details

a) *Model configuration:* We query Claude Haiku 4.5 for every judgment in CoRS and the three baselines that employ foundation models, with no extended thinking and one sample per query. Claude Sonnet 5 writes the text of each region once from its observation images, and describes only the observation.

b) *CoRS implementation:* CoRS searches with A* [31], as do the baselines except Direct planning. The sort is a quicksort with a fixed seed, and it queries the model once for each pair it compares. The prompt names the regions by neutral labels, so the model reads only the texts. The search then sums the vector cost in (6) along the edges and orders the open list by it. The heuristic is the Euclidean distance to the goal on the component k_{len} and zero elsewhere, which is admissible and consistent.

c) *Baseline implementation:* Every planner reads the same instruction and the same texts of the regions. Direct planning makes one call to the model per problem. Candidate selection also makes one call and picks $m = 3$ routes farthest apart in Hausdorff distance among the routes through every place. Absolute scoring makes one call per region and scores it s from 1 to 5. The score scales the length inside the region

TABLE I
QUANTITATIVE BENCHMARK RESULTS OF THE BASELINES AND CoRS BY ENVIRONMENT

Planner	Behavior match $\uparrow$			Critical violation $\downarrow$			Over-conservatism $\downarrow$		
	Supermarket	Restaurant	Hospital	Supermarket	Restaurant	Hospital	Supermarket	Restaurant	Hospital
Shortest path	35.6	41.1	34.4	85.0	65.0	65.0	0.0	0.0	0.0
Uniform caution	44.4	44.4	44.4	25.0	25.0	25.0	66.7	66.7	66.7
Oracle ranking	66.7	66.7	66.7	0.0	0.0	0.0	66.7	66.7	66.7
Direct planning	38.4	40.4	34.0	58.0	52.0	64.5	20.0	24.7	5.3
Candidate selection	61.6	61.8	57.8	31.5	43.0	36.5	10.0	0.7	0.0
Absolute scoring	64.7	70.7	64.4	24.0	0.5	9.5	38.0	45.3	72.0
Ours: CoRS	76.0	79.6	81.3	6.5	0.0	0.0	36.0	30.0	48.7

All values in %. The planners with an LLM solve each arrangement under all five instructions, while the three without one solve it once. Best among the LLM-based planners in bold.

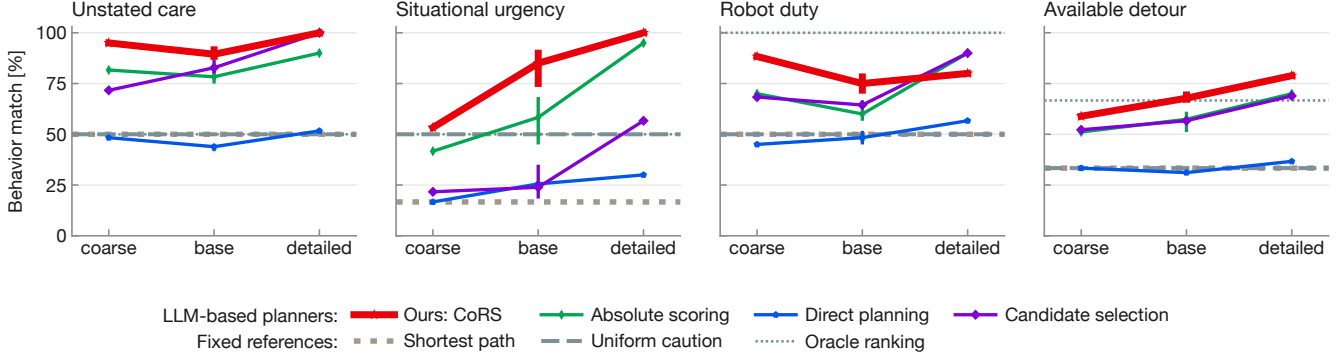


Fig. 4. Planner performance across the problem types and the levels of abstraction, in behavior match pooled over the three environments. At base, the marker is the mean across the three wordings, and the bar is their range.

by 6^z with $z = (s - 3)/2$, which adapts the cost of [22] to the longer edges of our map.

C. Results

a) Comparison with baselines: We first examine in Table I whether each planner avoids the right regions in each environment, neither too few nor too many. The three planners without an LLM mark the two extremes. Shortest path crosses a critical region in most of the problems, while Uniform caution and Oracle ranking take a needless detour in 66.7%. CoRS balances the regions against the length, so it violates a critical region in at most 6.5% and takes far fewer needless detours. As a result, it reaches the highest behavior match in every environment. Although Oracle ranking has the regions in the right order, its behavior match remains lower than CoRS, since it keeps every region above the length.

To answer Q1 and Q2, we examine Fig. 4, which shows behavior match by problem type and level of abstraction. In Unstated care, the coarse and base instructions leave out the regions to avoid, so a planner must choose a detour or a crossing from the situation alone. CoRS reaches the highest behavior match of 95% with the coarse instruction. It grounds the instruction on the observation and derives the consideration itself (Q1).

In the other three types, the situation stays the same while the deadline, the duty, or the detour changes the right route. A planner blind to these changes would return the same route in every condition. Behavior match is then at most 50%, or 33%

with the three conditions of Available detour. CoRS reaches a better behavior match in every type with every instruction. It ranks the regions and the length under each instruction, so the same arrangement returns a different route in each condition (Q2), as the case studies below show.

To answer Q3 and Q4, we compare CoRS in Table I with the three planners that employ an LLM. Direct planning has the lowest behavior match, since it writes an invalid route in 67% of the problems and falls back to the shortest path. Candidate selection remains low in Situational urgency even with the detailed instruction (Fig. 4), since its candidates come from geometry alone. Absolute scoring comes closest to CoRS, as it also turns the judgment into costs for planning. Yet its scores come out high and alike, too flat to order the regions. The planner then spends its detour on a region it could cross, and crosses the one worth a detour. CoRS thus returns the right route more often than a planner that writes or selects one (Q3). Its ranking also separates the regions that absolute scores leave alike (Q4).

b) Case studies: Fig. 5 shows three problem pairs, where the two panels differ in the remaining routes, the robot’s duty, or the task urgency. Each panel shows the routes of CoRS and the three baselines using an LLM, with the observation images below. In the hospital of Fig. 5a and 5d, the instruction says only “go to the goal.” CoRS avoids both regions when a route around them exists. Otherwise, it crosses the wet floor and keeps off the fallen hazardous box, which stands higher in the ranking. In the restaurant

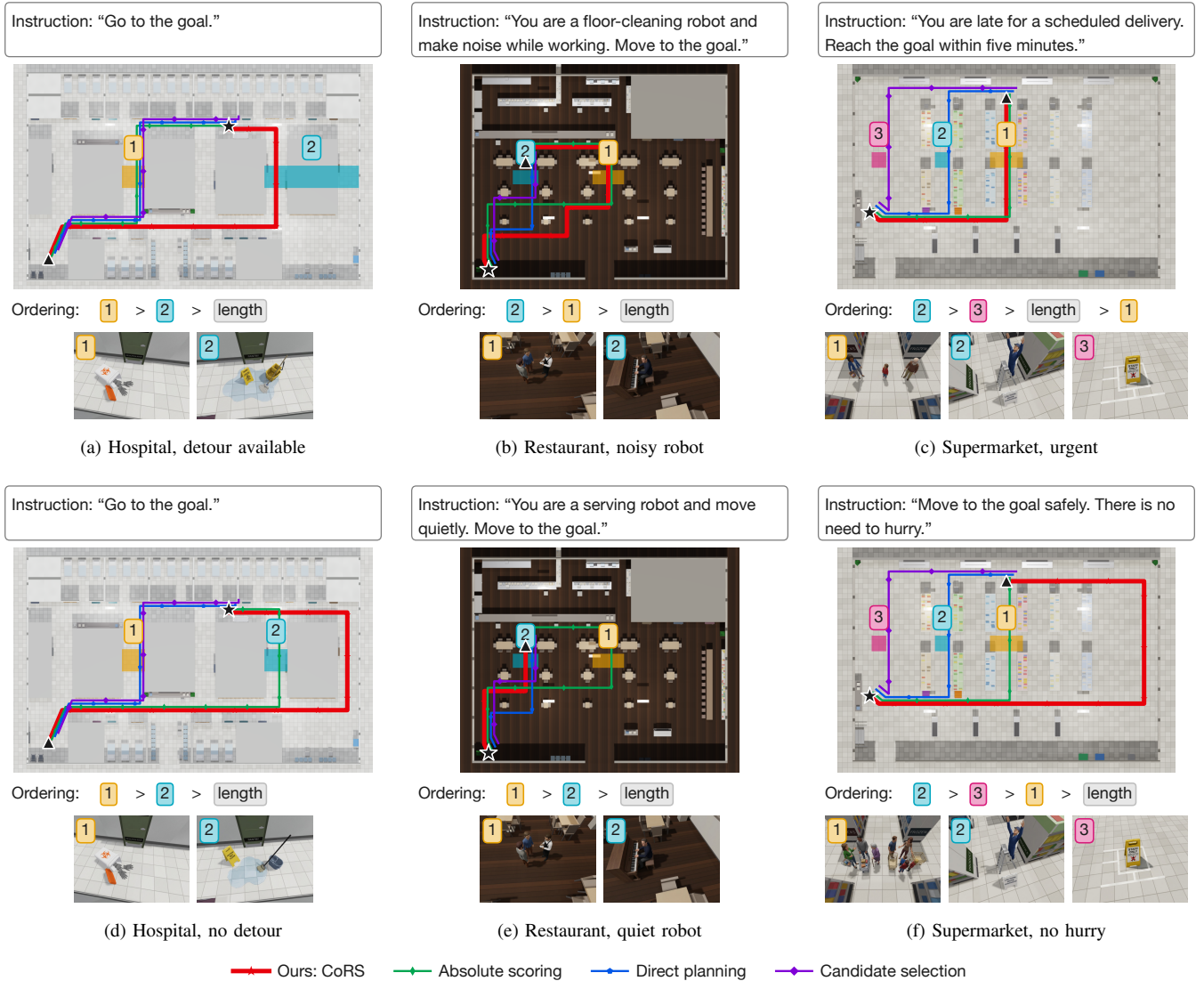


Fig. 5. Case studies. Each panel shows one problem, with the routes of the LLM-based planners over the map ($\blacktriangle$ start, $\star$ goal), the ranking CoRS returned, and the image of each numbered region. The two panels in a column share the start and the goal. Where routes coincide, the baselines are drawn slightly offset; CoRS is drawn on the map’s own geometry.

of Fig. 5b and 5e, the duty in the instruction changes from cleaning with noise to serving quietly. The noisy robot then avoids the person playing the piano and passes the two customers talking with a waiter, while the quiet robot does the opposite. In the supermarket of Fig. 5c and 5f, the instruction changes from an urgent delivery to no hurry. CoRS avoids the worker on a ladder and the staff-only area under both instructions. The robot instead passes the customers in the aisle to arrive in time, while it takes a longer route that avoids them without the deadline.

c) Ablations: We run ablations of CoRS in Table II, which replace one part of the judgment. Source descriptions give the LLM the texts we wrote to render the images without a VLM. The rest vary the thinking budget and the model, which both shape the quality of each judgment. For the ablations, we also read the pairwise judgments behind the

TABLE II
ABLATION OF CoRS OVER ALL ENVIRONMENTS

	Behavior match $\uparrow$	Pair agreement $\uparrow$	Over- conservatism $\downarrow$
Ours: CoRS	79.0	75.5	38.2
Source descriptions	95.3	91.6	5.6
Low thinking	77.2	78.0	40.3
High thinking	76.3	75.1	40.7
Sonnet 5	80.8	79.2	17.6
Opus 5	76.3	89.6	42.5

All values in %. CoRS uses Haiku 4.5 for the judgments, as in Table I. The rows below replace the descriptions, the thinking budget, and the judgment model in turn.

ranking in Section V-D. Pair agreement is the fraction of the comparisons that agree with them.

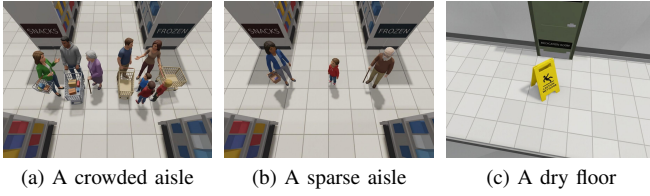


Fig. 6. Example observations where the description misses a fact about the space around the objects on the floor.

We first look at source descriptions, which give the largest gain. While VLM-generated descriptions remain neutral on the judgment as intended, they often miss the facts behind one. For example, they capture the people and carts blocking an aisle (Fig. 6a), but miss the navigable gaps between shoppers (Fig. 6b). They also read a cleaning sign but not the dry tiles under it (Fig. 6c), so CoRS reads the sign as a wet floor and detours, while the source text crosses.

A stronger model and a larger thinking budget both supply the hazards the description leaves out. On the same dry floor, Opus 5 assumes the description misses a film of water too thin to see and takes a detour twice as long. It agrees with our reference best, yet it detours around 91% of the regions, against 82% for CoRS. A larger thinking budget turns the judgment from the people on the floor to the robot’s own safety. In one problem, the robot avoids a puddle rather than a person crouching beside a fallen cane.

D. Limitations and Future Directions

Although CoRS takes far fewer needless detours than Absolute scoring, it still has room to rank a region against the detour around it. As the ablations suggest, a stronger model would plan better if it did not assume beyond the observation. Another limit is that CoRS judges the regions once before the robot moves, so a situation that changes on the way falls outside the ranking. Running the judgment online would need local observations, and the ranking could then feed the lower-level controller.

VII. CONCLUSION

We presented CoRS, a path planner that turns an abstract instruction into a route that follows commonsense. It infers the latent considerations from each region, ranks the regions against one another, and searches for a valid route on the resulting costs. The same region is thus avoided in one situation and crossed in another, as the instruction, the task, and the remaining routes change. On a benchmark of simulated everyday environments, CoRS followed commonsense more often than every baseline. Future work will integrate CoRS into a navigation system with local observation and control, and deploy it in the real world.

REFERENCES

- [1] K. Weerakoon *et al.*, “BehAV: Behavioral rule guided autonomy using VLMs for robot navigation in outdoor scenes,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 7044–7051.
- [2] D. Huo, J. Wang, C. Gao, Y. Qiao, D. Zhang, and G. Zhou, “NORM-Nav: Zero-shot mobile robot navigation with natural language behavioral constraints,” arXiv:2605.16979, 2026.
- [3] J. Bao and R. Yonetani, “Path planning using instruction-guided probabilistic roadmaps,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 505–511.
- [4] D. Shah, B. Osinski, B. Ichter, and S. Levine, “LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action,” in *Proc. Conf. Robot Learn.*, 2022, pp. 492–504.
- [5] C. Huang, O. Mees, A. Zeng, and W. Burgard, “Visual language maps for robot navigation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 10608–10615.
- [6] J. Tümová, L. I. Reyes Castro, S. Karaman, E. Frazzoli, and D. Rus, “Minimum-violation LTL planning with conflicting specifications,” in *Proc. Amer. Control Conf.*, 2013, pp. 200–205.
- [7] J. Tümová, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, “Least-violating control strategy synthesis with safety rules,” in *Proc. ACM Int. Conf. Hybrid Syst.: Comput. Control*, 2013, pp. 1–10.
- [8] L. I. Reyes Castro, P. Chaudhari, J. Tümová, S. Karaman, E. Frazzoli, and D. Rus, “Incremental sampling-based algorithm for minimum-violation motion planning,” in *Proc. IEEE Conf. Decis. Control*, 2013, pp. 3217–3224.
- [9] A. Censi *et al.*, “Liability, ethics, and culture-aware behavior specification using rulebooks,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 8536–8542.
- [10] Z. Fang, A. Xiao, D. Hsu, and G. H. Lee, “From obstacles to etiquette: Robot social navigation with VLM-informed path selection,” *IEEE Robot. Autom. Lett.*, vol. 11, no. 4, pp. 3947–3954, 2026.
- [11] J. Chen, B. Lin, R. Xu, Z. Chai, X. Liang, and K.-Y. Wong, “MapGPT: Map-guided prompting with adaptive path planning for vision-and-language navigation,” in *Proc. Annu. Meeting Assoc. Comput. Linguist.*, 2024, pp. 9796–9810.
- [12] G. Zhou, Y. Hong, and Q. Wu, “NavGPT: Explicit reasoning in vision-and-language navigation with large language models,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 7, 2024, pp. 7641–7649.
- [13] G. Zhou, Y. Hong, Z. Wang, X. E. Wang, and Q. Wu, “NavGPT-2: Unleashing navigational reasoning capability for large vision-language models,” in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 260–278.
- [14] S. Song, S. Kodagoda, A. Gunatilake, M. G. Carmichael, K. Thiagarajan, and J. Martin, “Guide-LLM: An embodied LLM agent and text-based topological map for robotic guidance of people with visual impairments,” in *Proc. Int. Conf. Sens. Technol.*, 2025, pp. 1–6.
- [15] A. J. Sathyamoorthy *et al.*, “CoNVOI: Context-aware navigation using vision language models in outdoor and indoor environments,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 13 837–13 844.
- [16] Z. Ravichandran, A. Robey, V. Kumar, G. J. Pappas, and H. Hassani, “Safety guardrails for LLM-enabled robots,” *IEEE Robot. Autom. Lett.*, vol. 11, no. 4, pp. 4649–4656, 2026.
- [17] S. Choi *et al.*, “CANVAS: Commonsense-aware navigation system for intuitive human-robot interaction,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 2445–2452.
- [18] Z. Chen *et al.*, “SocialNav: Training human-inspired foundation model for socially-aware embodied navigation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2026, pp. 28 796–28 806.
- [19] S. Zhao *et al.*, “NormAct: A benchmark for hidden social norm compliance in embodied planning,” arXiv:2606.27826, 2026.
- [20] D. Shah, M. R. Equi, B. Osinski, F. Xia, B. Ichter, and S. Levine, “Navigation with large language models: Semantic guesswork as a heuristic for planning,” in *Proc. Conf. Robot Learn.*, 2023, pp. 2683–2699.
- [21] D. Song, J. Liang, A. Payandeh, A. H. Raj, X. Xiao, and D. Manocha, “VLM-Social-Nav: Socially aware robot navigation through scoring using vision-language models,” *IEEE Robot. Autom. Lett.*, vol. 10, no. 1, pp. 508–515, 2025.
- [22] T. Sugino, S. Shiro, and K. Tanaka, “VANG-Planner: Integrating language instructions into spatial cost for global path planning,” in *Proc. IEEE Int. Conf. Robot. Autom. Workshop Multi-Modal Spatial Intell.*, 2026.
- [23] H. Zhang, H. Deng, J. Ou, and C. Feng, “Mitigating spatial hallucination in large language models for path planning via prompt engineering,” *Sci. Rep.*, vol. 15, no. 1, p. 8881, 2025.
- [24] Z. Qin *et al.*, “Large language models are effective text rankers with pairwise ranking prompting,” in *Findings Assoc. Comput. Linguist.*, 2024, pp. 1504–1518.

- [25] A. Liusie, P. Manakul, and M. Gales, “LLM comparative assessment: Zero-shot NLG evaluation through pairwise comparisons using large language models,” in *Proc. Conf. Eur. Chapter Assoc. Comput. Linguist.*, 2024, pp. 139–151.
- [26] M. Savva *et al.*, “Habitat: A platform for embodied AI research,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019.
- [27] S. K. Ramakrishnan *et al.*, “Habitat-matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI,” in *Proc. Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2021.
- [28] M. Deitke *et al.*, “ProcTHOR: Large-scale embodied AI using procedural generation,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 5982–5994.
- [29] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, “Soc-NavBench: A grounded simulation testing framework for evaluating social navigation,” *ACM Trans. Hum.-Robot Interact.*, vol. 11, no. 3, pp. 1–24, 2022.
- [30] J. Chen *et al.*, “LISN: Language-instructed social navigation with VLM-based controller modulating,” arXiv:2512.09920, 2025.
- [31] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.