VADER: Visual Affordance Detection and Error Recovery for Multi Robot Human Collaboration

†Michael Ahn[1] Montserrat Gonzalez Arenas[1] Matthew Bennice[2] Noah Brown[5] Christine Chan[1] Byron David[7] Anthony Francis[4] Gavin Gonzalez[6] Rainer Hessmer[2] Tomas Jackson[6] Nikhil J Joshi[1] Daniel Lam[2] Tsang-Wei Edward Lee[1] Alex Luong[6] Sharath Maddineni[1] Harsh Patel[2] Jodilyn Peralta[6] Jornell Quiambao[5] Diego Reyes[5] Rosario M Jauregui Ruano[6] Dorsa Sadigh[1] Pannag Sanketi[1] Leila Takayama[3] Pavel Vodenski[2] Fei Xia[1]

***Abstract*— Robots today can exploit the rich world knowledge of large language models to chain simple behavioral skills into long-horizon tasks. However, robots often get interrupted during long-horizon tasks due to primitive skill failures and dynamic environments. We propose VADER, a *plan, execute, detect* framework with *seeking help* as a new skill that enables robots to recover and complete long-horizon tasks with the help of humans or other robots. VADER leverages visual question answering (VQA) modules to detect *visual affordances* and recognize *execution errors*. It then generates prompts for a language model planner (LMP) which decides when to seek help from another robot or human to *recover from errors* in long-horizon task execution. We show the effectiveness of VADER through an experiment with a mobile manipulator asking for help from another mobile manipulator or another human for completing two long-horizon robotic tasks. Our user study with 19 participants suggests VADER is perceived to complete tasks more successfully than a control which does not ask for help, yet VADER is perceived as equally capable even though it receives human help. https://google-vader.github.io/.**

I. INTRODUCTION

Task and motion planning is a popular framework for solving long-horizon robotic tasks which treats high-level planning and low-level skills as interdependent [1], [2]. Recently, language model planners (LMPs), which use large language models (LLMs) to orchestrate a library of low-level *skill* primitives, have shown promising results in replacing traditional task planners [3], [4]. LMPs exploit LLM's rich semantic structure to generate high-level robot plans given diverse human instructions. For example, LMP-based robots can execute plans for instructions like *"get me a drink"* or *"I am thirsty, please help me"*, issued in many languages, even if they have not seen this exact instruction in *any* language, by using LLM knowledge to break these tasks into skills and perform them in the correct causal order.

While LMPs alleviate the need for task planning modules, their language plans are often not fully grounded in the robot's environment [5], which can make it hard to evaluate whether a skill should be executed (skill affordances) or whether a skill has succeeded (error recognition). Incorrectly gauging skill affordances can cause planning errors leading to execution of incorrect skills. In addition, skill failures can disrupt task execution. These issues compound as task horizons become longer: ironically, as a robot becomes more capable, it can fail more often!

For example, LMPs such as SayCan [3] ground language plans in skill affordances that are based on value functions associated with an RL policy used for executing each skill. While this promotes selecting a plan with skills with high affordances, LMPs often lack awareness of the current state. If a robot breaks its gripper or its workspace is disrupted, many LMPs cannot recognize these dynamic changes and might proceed with an infeasible plan. Furthermore, while LMPs use affordances as pre-conditions for selecting the *next* skill, they generally assume success at previous skill execution lacking the ability to detect errors.

Recent efforts attempt to address these issues by bringing environmental cues, such as scene descriptions into the planning loop. For example, extensions of SayCan such as Inner Monologue [6] improve reliability of execution by incorporating a variety of environment feedback into the LMP planning loop. However, while feedback has been shown to be effective, the existing systems typically focus on failures that can be resolved by a single robot. If the problem cannot be resolved given the robot's own capabilities – e.g., a robot gripper breaking – task execution will still fail.

Our key insight is that by grounding with their environments, robots can detect their failures and collaborate with other robots, and humans to course correct through planning. Today's visual question answering (VQA) systems [7] can provide the required grounding mechanism, where natural language summary on a visual observation can be generated within the context of a query. But a multi robot human collaboration is hard due to lack of a mechanism for distributed communication that enables agents to post or claim tasks and provide assistance to each other.

Concretely, our contributions are: (a) a general-purpose technique called *Visual Affordance Detection and Error Recovery* (VADER) which uses feedback from affordance and error detection to generate requests for help from other agents or humans (Fig. 1). This allows the system to dynamically detect failures and employ recovery measures, thus enabling it to complete long horizon tasks. (b) a cloud-based communication framework to facilitate this assistive collaboration, instantiated with robot agents working alongside humans.

† Authors are listed in alphabetical order. [1]Robotics at Google [2]Everyday Robots [3]Hoku Labs [4]Logical Robotics; work begun while at Robotics at Google [5]FS Studio [6]Relentless Adrenalin [7]MoBack

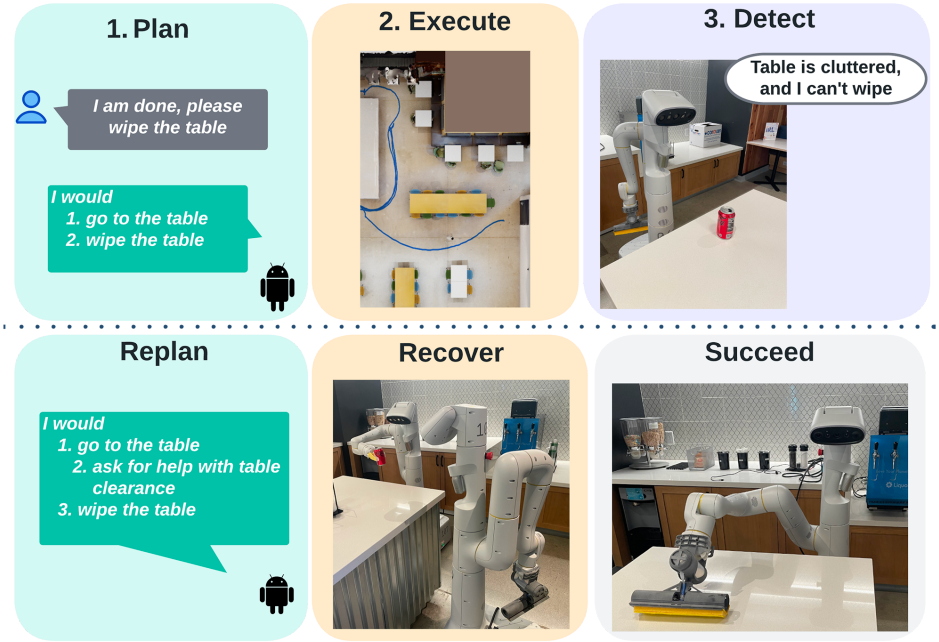

Fig. 1: **VADER: Visual Affordance Detection and Error Recovery**. A *plan, execution, detect* framework with a *seeking help* skill as a recovery mechanism. While executing a plan, the robot detects a deviation from its expectations – a coke can obstructing the table area to be wiped. It replans for recovery, and upon receiving help, completes the original plan.

(c) demonstration of the effectiveness of our approach on a complex, long horizon task, where two robots with differing morphologies (one with a parallel gripper and one with a wiping tool at its end-effector as shown in Fig. 3) have to work in collaboration to complete, i.e. the task cannot be done by any one of them alone. (d) a user study with 19 participants in office kitchen spaces in which VADER not only was more successful at completing a long-horizon task than a control which did not ask for help, but also appeared more effective according to human users.

II. BACKGROUND

Prior work on Multi Human-Robot Collaboration. There is a plethora of work on both multi-robot and human-robot communication and collaboration [8], [9]. Multi-robot coordination often considers task allocation in settings such as collaborative manipulation [10], [11], UAV formation, and multi-agent search and rescue [12], [13]. While these works are limited to collaboration between robots, prior work also considers effective human-robot collaboration [14]. This includes shared autonomy settings that arbitrate or blend human and robot inputs [15], approaches towards partner modeling [16], [17], as well as robots asking for help from nearby people [18], [19], [20], [21], [22]. To the best of our knowledge, prior work does not consider a multi-human-robot collaboration framework that can detect each agent's affordances and effectively recover by asking for help.

Low-Level Skills. A general robot-environment interaction can be modeled as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$ defined over the state space $\mathcal{S}$ capturing the environment and robot state, the robot action space $\mathcal{A}$, the transition probability $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with a discount factor γ. Executing a *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ in an environment results in a trajectory, i.e., a sequence of states, actions, and rewards $\tau = \{(s_0, a_0, r_0), ..., (s_N, a_N, r_N)\}$. Our goal is to create an execution policy that optimizes its expected discounted accumulated reward, or *return* $G = \mathbb{E}_{\tau \sim \pi}[\sum_{k=0}^{N} \gamma^k r_k]$ [23] over the trajectories that result from following the policy. In reinforcement learning (RL) the policy is learned over the returns received in each trajectory, whereas in behavioral cloning (BC) the policy is learned over expert trajectories that implicitly optimize the return. In model predictive control (MPC), the policy implicitly attempts to achieve high returns by optimizing a proxy *cost function* over forward rollouts of trajectories based on the state-action transition probability P.

A *skill* σ refers to a sensorimotor primitive described in natural language as defined in [1], [3]. Every skill σ_i has an associated policy π_i used for executing it. In this work, no skill learning was involved: we assume access to a library of precomputed policies. In addition, these skills are not limited to come from the same source and can be created independently. For example, our manipulation policy was trained using BC with a transformer based architecture [24], our table wiping skill was trained with RL [25], and our navigation skill uses the non-learned MPC baseline from [26].

Language Model Planners. Language model planners (LMPs) combine the semantic and causal structure of the world embedded in an LLM with the skills acquired by a robot to construct task execution plans from available skills. Language model planning involves using a language model to transform a task instruction $\mathcal{T}$ into a larger plan $\mathcal{P}$ consist-

Algorithm 1 VADER

Given: A set of agents $\mathcal{A}^{(j)}$, a set of skills $\Sigma^{(j)}$ with associated execution policies $\Pi^{(j)}$, an outcome description functions $\mathcal{O}_\Sigma$, a task-instruction λ, and environmental state s_0

let c_n be the task execution context update at planning step n.
$n = 1, c_0 = \emptyset$
while $c_{n-1} \neq$ "done" **do**
*# **1. Plan**: Execution Skill selection (including "ask for help") using LMP*
    $\sigma_n = \arg\max_\sigma p^{\text{LMP}}(\sigma|\lambda, s_n, c_{n-1}, ..., c_0)$
 # Description of successful/failure outcomes of π_σ
    $\ell^{\text{exp}}_{n+1} = \mathcal{O}(\sigma_n, s_n)$
*# **2. Execute**: the skill by the robot itself or ask for help from an external agent. State evolves to s_{n+1}.*
    execute $\pi^{(j)}_n(s_n)$ in the environment
*# **3. Detect**: VQA context affordance*
    $\ell^{\text{assess}}_{n+1} = \arg\max_\ell p^{\mathcal{V}_{QA}}(\ell|\ell^{\text{exp}}_{n+1}, s_{n+1})$
 # LMP context update for replanning
    $c_n = concat(\sigma_n, \ell^{\text{exp}}_{n+1}, \ell^{\text{assess}}_{n+1})$
    $n = n + 1$
end while

ing of a sequence of executable skills as defined earlier. For example, the language model can be used to score a fixed set of language representations of available skills Σ to produce a ranking for the next skill to be executed σ_n in the plan. In [3] the LLM ranking scores were "grounded" [5] by multiplying them with the value functions $V(s)$ associated with the skills as a proxy for their affordances $p^{\text{affordance}}(\sigma|s)$ to obtain what we collectively denote as $p^{\text{LMP}}(\sigma|\mathcal{T}, s, \sigma_{n-1}, ..., \sigma_0)$. The skill σ that maximizes p^{LMP} is typically selected to be executed next in the task plan. We use PaLM [27] as the language model for task planning in our work in a similar fashion as prior work such as [3].

Visual Question Answering. Visual-Language Models (VLM) leverage the abundant (image, text) paired data to learn bi-encoders that map texts and images to the same embedding space E in an attempt to capture semantics and transfer concepts across these two modalities [28], [29], [7]. VLMs such as CLIP [28] show promising zero-shot classification capabilities to novel concepts based on these encodings, while other VLMs such as ViLD [29] distill the vision-text knowledge into open vocabulary object detection and mask prediction models. Recently PaLI [7] leveraged pretrained LLMs with relatively moderate sized vision models to transfer generalization capabilities acquired by former to the latter. We experimented with all three of these VLM variants in this work, using them as vision question answering (VQA) systems to answer text queries about images with text answers, $\mathcal{V}_{QA} : \mathcal{I} \times \mathcal{T} \to \mathcal{T}$.

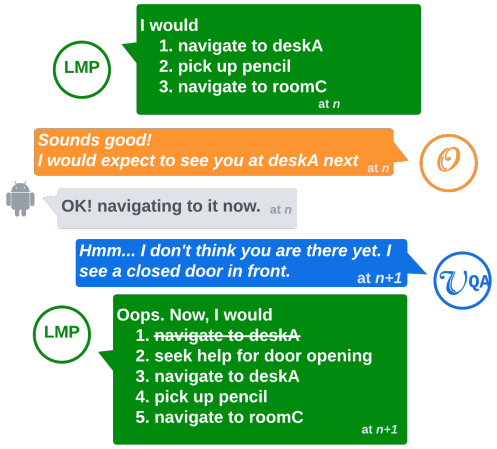

Fig. 2: LMP Replanning with Request for help. The process is imagined as a conversation between different components. The immediate next skill from original LMP plan at step n is passed through outcome description function $\mathcal{O}$ and the $\mathcal{V}_{QA}$ for execution status assessment and folded back to LMP. A recovery plan is laid out.

III. VADER

In a nutshell, the problem we are interested in addressing is given a long-horizon task instruction λ in natural language such as *"I am done, please wipe the table"* as in Fig. 1, and a library of low-level skills present to a robot Σ, we would like the robot to generate a task and motion plan that effectively executes the long-horizon task λ.

We introduce Visual Affordance Detection and Error Recovery or VADER, a *plan, execute, detect* framework, defined in Algorithm 1, which uses a VQA to verify the execution of each LMP skill before continuing, similar to the principle of the Test-Operate-Test-Exit loop [30] – but adding the key ability to recover within this loop with the help of other agents. This is similar to recovery from failure by asking for help in [31], though that work asked help only from humans, using a classical planner [14] rather than an LMP and a custom request generator rather than a VQA.

The key insight of VADER is instead of executing the task plan given the library of skills Σ in an open-loop fashion, the robot should detect its visual affordances and perform error recovery by replanning and asking for help from other agents. Specifically VADER traverses the loop in Fig. 1:

1) VADER first *plans* the next primitive to be executed by selecting the skill σ from the library of skills Σ with the highest probability for success. For instance, the robot might decide on selecting the skill σ = "pick up coke can". We simultaneously generate a language description ℓ^{exp} = "coke can in hand" for the expected desired future state if the execution was successful.
2) We then *execute* the skill, resulting in a new state s_{n+1} when starting from s_n.
3) We *detect* any deviations or errors by comparing the new state s_{n+1} against the expected description ℓ^{exp}.

Deviations could include failures in skill execution, losing

the capacity for executing the skill (for example, breaking a gripper), or even incorrectly picking up tasks it can't complete given its skillset. In the case of skill failure, replanning (which occurs during the *plan* and *execute* phases) may bring the robot back to the desired state, as in [6]. In the case of loss of capacity or claiming an out-of-scope task, however, replanning may not be feasible: if unable to complete the task on its own, the agent may need to seek help from another robot or a human to recover (which occurs during the *detect* phase). VADER thus enables a cooperative environment of robots with diverse skills sets or even morphologies sharing an ecosystem with humans as shown in Fig. 1.

To make this idea concrete, henceforth we assume coexistence of agent variants $\mathcal{A}^{(j)}$'s with diverse skill specializations $\Sigma^{(j)}$. For example, as shown in Fig. 1 a robot might have a gripper being able to pick up a coke can while another robot only has a wiping tool at its end-effector allowing it to perform other types of skills such as wiping a table. We assume humans have the ability to perform any of the skills needed for long-horizon tasks. However, to minimally disrupt the autonomy of the overall robotic system, the VADER algorithm prioritizes asking for help from other robots before asking for human intervention.

Task execution failures that VADER aims to recover from can be grouped broadly according to their causes:

- **Infeasible States.** This may be the most common point of failure. For example, a robot may break its gripper, or be blocked on its path, or subscribe to a task requiring a skill outside its capabilities.
- **Skill Execution** *thinks* **it was Successful, but Failed**. For example, while a robot is cleaning a table it might drop the debris due to a poor grip, but the manipulation policy may still finish normally.
- **Skill Execution Fails and Alerts.** In rare cases, the skill execution policies are themselves able to sense failures in its execution and halt. For example, a navigation policy may declare infeasible in reach after exhausting all possible paths of approach unsuccessfully.
- **Erroneous planning.** A wrong planning step chosen by LMP can put the execution on an undesirable path eventually making the robot fail. However, because VADER relies on the LMP for planning in the loop and has no access to the execution history or broader context, it cannot detect failures in planning.

VADER adds three key components to an LMP to enable recovery from failures: (a) *detection* of skill affordances and execution errors with visual question answering, (b) *replanning* based on the detected categories of failures, and (c) *recovery* based on seeking help from other agents.

Note steps (a) and (b) are similar to the closed-loop feedback proposed in [6], with the key innovations in this work being that in (a) we also check for skill affordance failures where a robot has taken on a skill it either cannot or has become unable to perform; detecting these out-of-scope failures informs the replanning choices in (b) to consider assistance from other agents via (c). Another key difference is that asking for help on a failed *skill* can result in an entirely new *task*, which itself may require the execution of several skills for its completion. For instance, the wiping robot in Fig. 1 might realize it can't wipe when there is a coke can on the table. Asking for help will lead to another robot performing a long-horizon task of de-cluttering the table by navigating to it and picking up the coke can.

A. Detection: *VQA for Affordance Evaluation*

The main purpose of VQA in VADER is to estimate skill affordances in a closed-loop, policy-agnostic way – as opposed to estimating skill affordances based on RL policy value functions as in [3], which are mainly used for open-loop skill selection during planning, prior to execution, and may not be trivially available for BC or MPC. From the perspective of VADER, we consider value function affordances, if used at all, to be implicitly embedded in p^{LMP}.

VQA-based affordance detection can be applied in a plug-and-play fashion using the current state and a language representation of the expected outcomes of the executed skill ℓ^{exp}. We denote the skill *outcome* description function responsible for generating natural language descriptions of the possible states s_{n+1} given successful or unsuccessful execution of a skill by $\mathcal{O} : \Sigma \times S \rightarrow \Lambda$. In its simplest form, $\mathcal{O}$ could be a lookup table from skill descriptions to outcome descriptions. However, the LMP could be modified to output both the selected skill σ_n to be executed at s_n and the expected outcome ℓ^{exp}_{n+1} at time step s_{n+1}. If knowledge of the current state is required for this assessment, the same VQA used for estimating skill affordances can be used for assessing the expected outcome ℓ^{exp}_{n+1} by comparing the skill description σ_n with the current state s_n.

From the perspective of VADER, the output of VQA is a language assessment of the last skill execution $\ell^{\text{assess}}_{n+1}$ which is appended to the skill σ_n and fed back to the LMP. In its simplest form, $\ell^{\text{assess}}_{n+1}$ could be computed over a fixed set of expected outcomes of skill execution over which we pick the answer with the maximum score $\arg\max_\ell p^{\mathcal{V}_{QA}}(\ell, s_{n+1})$, as in the case of zero-shot VQA based on ViLD or CLIP. Alternately, an open-set VQA system like PaLI could compute $\ell^{\text{assess}}_{n+1}$ from the current state s_{n+1} based on a text query based on the expected outcomes ℓ^{exp}_{n+1}.

For error detection, in the navigation example, the VQA prompt could be *"is the robot at posA"* applied to an image of the current localization state, and expected outcomes might include "$\{\text{no}, \text{yes}\}$" with "yes" denoting success. If the answers are scored "$\{\text{no} : 0.55, \text{yes} : 0.32, ...\}$" then the assessed execution success of "navigate to destination posA" $\ell^{\text{assess}}_{n+1}$ would be "no", resulting on the LMP replanning to recover from failure on the next step.

For affordance detection, we need to check the precondition of an affordance prior to execution; this is performed by novel *information-gathering skills* that we have created and which the LMP can use to check the preconditions of the next skill. For example, a wiping robot may need the table to be clear of clutter prior to being wiped. Therefore, the LMP may break the task "*wipe the table*" into the skills "*drive to the table*", "*check if the table is clear*", and "*perform*

table wiping". The skill "*check if the table is clear*" is an information gathering skill that checks the prerequisites for "*perform table wiping*" by looking at the table. ℓ_{n+1}^{exp} for this skill may be "*is the table clear for wiping?*" applied to the current camera image using an open-set VQA system like PaLI. If the answer is "no" then the prerequisite for the "*perform table wiping*" skill would fail and the LMP would replan by asking another agent to remove the clutter.

B. Replanning: *Absorbing Failures in the LMP*

The previously selected skill σ_n is appended with the outcome expectation ℓ_{n+1}^{exp} and the execution success assessment $\ell_{n+1}^{\text{assess}}$ to form a new *context* prompt $c_n := concat(\sigma_n, \ell_{n+1}^{\text{exp}}, \ell_{n+1}^{\text{assess}})$ that is fed back to the LMP for replanning. In the earlier example of the LMP planned skill of *"navigate to destination posA"*, failure would result in the context prompt as *"navigate to destination posA. at posA? no"*. The LMP would generate a new plan as shown in Fig. 2.

C. Recovery: *Seeking Help*

For a robot to be a useful, autonomous assistant, it needs ways to recover from failures, preferably without intervention from the original task requester – but that does *not* mean that the robot cannot request help. While VADER can handle cases where a robot can recover by retrying the same task itself, as in [6], in many practical scenarios this is neither desirable nor feasible. Instead, in VADER a robot which halts during task execution can request help from a nearby human or from another robot. While we assume humans are so skilled that the environment always affords them completing any task, in order to preserve autonomy of the overall robotic system in aggregation, VADER prefers receiving help from another robot before asking a nearby human for help.

Vision Models For Human Detection. To seek help from a nearby human, VADER relies on the robot's native hardware and software capabilities for human entity and depth perception and does not assume any specific dependency. VADER also does not assume human intent prediction capabilities essential for effective social interaction in crowded spaces. In this work, we always pick the nearest human to request help from. Note, even if a robot cannot detect nearby humans, it can use the Human Robot Fleet Orchestration Service (HRFS) defined in the next section to ask for help.

Human Robot Fleet Orchestration Service. We only ask nearby humans for help, but when seeking help from another robot VADER does not assume it is physically close. Also, a robot currently asking for help from others may later be accepting a request coming from elsewhere. To facilitate communication between a large fleet of robots we introduce Human Robot Fleet Orchestration Service (HRFS).

HRFS is a scalable, cloud-based, plug-and-play, real-time transactional communication service supporting multimodal communication among its participants. Agents can post tasks which can be claimed by other agents. For example, a robot may push a task like *"open the door"*, potentially with an executor preference of *"human"*. HRFS is agent agnostic and can accommodate a robot fleet of any size and variety, and many humans (teleoperators or in-person).

IV. EXPERIMENTS

We design our experiment and perform analysis to answer the following questions: **1.** whether VADER can enable robots to complete complex, long horizontal tasks by detecting failures plus cooperating with each other or humans, better than a robot that does not ask for help, and **2.** whether the robot is perceived to be less capable if it asks for help compared to another that does not ask for help.

A. Experiment Setup

We use two kinds of robots in our experiments. The **manipulation expert** (ME) has a gripper attached to its arm suitable for executing pick-n-place tasks (Fig. 3 middle top), while the the **table wiping expert** (TW), has a specialized tool for wiping table surfaces (Fig. 3 middle bottom).

A common office kitchen room with a beverage and snack area, including some chairs and tables steps away from the snack area is used for conducting all the experiments.

We first demonstrate our approach with a complex, long-horizon task that two robots with different morphologies achieve by working in collaboration. Note that neither of the robots is capable of finishing the task alone. In Sec. IV-D, we will discuss human-robot collaboration.

We use a setup as shown in Fig. 3. The two robots are parked in the north end of the office kitchen, and are tasked to clean up a table located at the south-west end of the kitchen (annotated as *Table wiping site* in Fig. 3). Tasks used were one of *"wipe the table"* – executable only by the **table wiping expert** – or *"clear the table"* – executable by only **manipulation expert**. Our system can detect a nearby human [32] and request help in human understandable, natural language voice prompts (generated by the LMP).

Simulating common blockages, both the table wiping site and the navigation routes to the site are blocked with obstacles like a *coke can* in former and *chairs* in the latter. This enforces collaboration between two robots as well as human intervention to successfully complete the task.

B. Implementation Details

For HRFS implementation, we used Firebase Realtime Database (RTD), which provides all the HRFS features needed for our experiments. We deliberately make our robots unaware of their morphological differences – access to gripper or wiping tool – at the initiation. As a result, any task posted to HRFS could be picked up by any of the robots. We expect the robots to detect its tooling capabilities (gripper vs wiping tool) upon a task assignment and update the execution plan accordingly, i.e., return the task to HRFS without further executing *or* proceed with the task.

We use PaLM [27] based setup similar to [3] for LMP. For $\mathcal{V}_{QA}$, we tried three state-of-the-art models , namely CLIP [28], ViLD [29], and PaLI [7]. They have different trade-offs between inference speed and accuracy. But in our experiments, the difference in performance is minor. We used simple lookup tables for the outcome function $\mathcal{O}$.

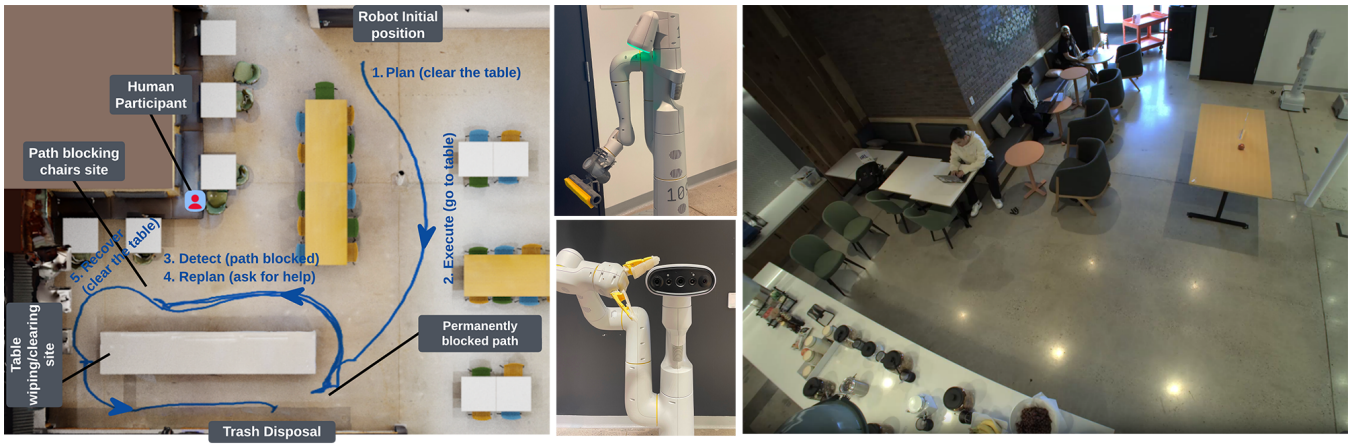

Fig. 3: **Experimental Setup. Left** The experimental setup with robot trajectory overlaid onto the *matterport* scanned version of the space used for experiments. **Middle.** (top) **table wiping expert** with the wiping tool, (bottom) **manipulation expert** with gripper. **Right.** A snapshot from one of experiments from our user study discussed in sec. IV-D.

C. Robot Collaboration Pilot Study

We setup a scenario commonly encountered in offices or homes: an obstructed space occluding task execution. The TW was tasked to *"wipe the table"* where the table was deliberately cluttered with obstacles like a *coke can* that obstructs the wiping path. VADER empowers TW to seek help from ME to clear the table before further wiping.

To spice up the challenge, our robots were not told whether they were TW or ME, and also had access to the exact same set of skills (i.e. $\Sigma^{(\mathrm{TW})} = \Sigma^{(\mathrm{ME})} = \{\mathrm{table-wiping}, \mathrm{manipulation}, \mathrm{navigation}, \mathrm{inspection}\}$), even when each could perform only a subset of the skills meaningfully. As a result, upon receiving a task they were required to detect context on their variety before proceeding with execution, a skill we called *Self-Inspection*.

The success rate of self-variant classification was 90%, while that for table clutter detection (one of the information gathering skills discussed in sec. III-A) was 98%. The two robots completed the task in 6 out of 11 trials, 4 out of which were due to robot hardware failures and 1 due to clutter detection failures. Inter-robot communication failed once due to an RTD update failure, but it recovered noninvasively after a network re-connection.

A successful episode from this pilot study can be found on https://google-vader.github.io/. The episode ran 11 minutes end-to-end. The first 3 minutes were spent from TW claiming the wiping task up to requesting help, the next 4 minutes in ME clearing the table while TW waited on an update from ME, and the last 4 in TE completing table-wiping while ME simultaneously put the coke can in the trash.

D. User Study

While the pilot study confirmed viability of VADER with two robots collaborating on completing table-wiping task, to further assess the effectiveness of VADER, we conducted a human-robot interaction study to compare how people would respond to each of two different versions of possible robot behaviors – (1) not asking for help (control condition), and (2) asking for help autonomously (VADER).

We focused our user study on addressing whether the robot is perceived to be less capable if it asks for help compared to another that does not ask for help. This is a pressing question because it is possible that robots asking for help from people will make those robots seem to be less competent compared to robots that simply give up on performing the task. On the other hand, people may perceive robots asking for help as more collaborative and useful.

1) HRI Study Design: To address our research question (above), we ran a within-subjects experiment in which we asked each participant to come into an office kitchen area and interact with a robot three times – once in each of the three experiment conditions. We counter-balanced the study for order of presentation of the conditions. A total of 19 volunteers participated in our study, but one of them failed to complete the full set of questionnaires so we did not include their data in the statistical analyses.

We selected participants neither familiar with our testing robots nor frequent users of robotics systems. Upon arrival, we explained to the participants the setup and the robot HRI interface. The robot was given the goal to *"clear the table"* as the high-level task, where the route to the manipulation site was blocked with two chairs (Fig. 3 (left)). At the beginning of each round, the participant was asked to stand near the snack area and wait for the robot to move. Depending on the condition (control or VADER) the robot may ask the participant for help completing the task.

After each session, we asked the participant to fill out a brief questionnaire, responding on a 7-point Likert scale to: **1.** "The robot asked for help in a timely fashion when help was needed", **2.** "I was able to understand when the robot needed help", **3.** "I was able to successfully help the robot", **4.** "The robot was able to continue with the task after me helping with part of the task", **5.** "The robot was successful at accomplishing the task it was asked to do", **6.** "I feel like I can trust the robot to accomplish the task", **7.** "I feel like I

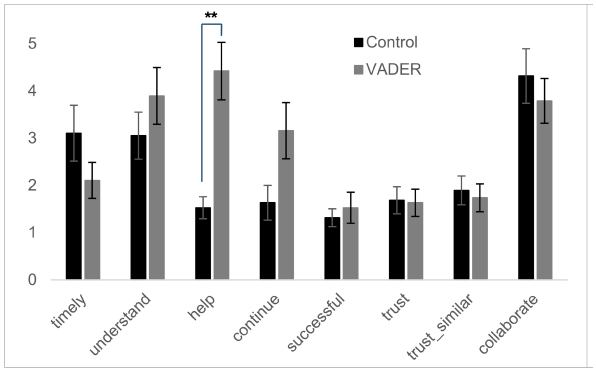

Fig. 4: Means and standard errors for participant responses to HRI questionnaires. VADER outperformed control significantly on the statement where participants were asked if they could help the robot successfully. No significant difference was observed between the conditions on other statements.

can trust the robot to accomplish other similar tasks", **8.** "I would like to collaborate with this robot in the future".

2) HRI Study Results: To evaluate how VADER compared to the control condition (the robot not asking for help), we ran a repeated measures analysis of variance (ANOVA) with all levels of the independent variable (VADER, control). Because we asked 8 questionnaire items, we used a Bonferroni correction [33] to adjust the p-value cut-off for statistical significance, .05 / 8 = .006. We report upon the pairwise contrasts of VADER vs. Control. Fig. 4 shows our participants found VADER significantly more effective than the control on 1 out of 8 statements. People agreed more strongly to the statement that they were able to help the robot running VADER (M=4.42, SE=0.61) than when the robot did not ask for help (control condition) (M=1.53, SE=0.23), pairwise contrast $F(1,12)=13.78$, $p=.003 < .006$.

In all other statements, we did not observe statistically significant differences between VADER and the control.

V. DISCUSSION

Interpretation. Our pilot robot collaboration study demonstrated that VADER was *viable* as an approach for performing long-horizon robotic tasks using robot-robot cooperation to recover from affordance failures. Our user study expanded the results by showing that people could be incorporated into the VADER framework with little to no changes, also enabling recovering from execution errors. While asking for help might be perceived as incompetent, our user study shows that it is not the case with VADER. Furthermore, VADER significantly improved the success rate because it enabled robot-robot and robot-human collaboration.

A preliminary analysis of the task-execution data reveals that our end-to-end episode with VADER condition took about 3-4$\times$ longer ($>$ 15 vs 5 minutes for the control). More than 65% of the time was spent by the robot in waiting for LMP responses running on cloud (as they could not be deployed on the robot hardware due to their sizes).

Robots using VADER were always able to complete the task in our user study. However, they scored low on *timely* aspect (Fig. 4) compared to the control, which failed fast. Our hypothesis is that a robot able to recover is useful, but it should act in a timely manner to be desirable as an assistant.

Limitations and Future Work. While our work shows that VADER enables robots using LMPs to use visual cues to ask for help from robots or humans, the presented approach has a number of limitations, some of which suggest future work.

- **Study design:** Our study design of users continuously observing the robots until they fail and ask for help does not well represent VADER's target scenario, in which humans are approached only when required. We will design future studies to better align with this scenario.
- **Helper determination:** In our study, both humans and robots can respond to tasks. Future work could recommend which agent can better provide help.
- **Help via dialog:** The LMP uses one voice prompt to request help, but humans may need more context to understand the task. Using dialog-based methods such as Google's Bard [34] or ChatGPT [35] could improve the likelihood of success.
- **Respecting social norms:** Our robots search for the nearest person for help. This could be suboptimal: for example, if the nearest person is in conversation, it will be better to ask another person for help.

We are excited about the possibility of building on our work to enable effective human robot collaboration with VADER to accelerate progress along the above fronts.

Conclusion. Large language models have shown the ability to plan over small skills and stitch them together into longer tasks, but this paradoxically has led to increased failure rates due to environment dynamics and skill brittleness. In this paper, we presented VADER, an approach which interleaves visual question answering-based error detection and recovery with help of other agents / humans into language model planning, thus allowing a team of humans and robots to achieve complex, long-horizon tasks.

ACKNOWLEDGMENTS

The authors would like to thank Carolina Parada, Jie Tan, Ben Jyenis, and Vincent Vanhoucke for their leadership, support and advice on the draft, as well as the robotics operations team at Google for their operators, wranglers and mechatronics engineering support and Everyday Robotics for support.

REFERENCES

[1] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *CoRR*, vol. abs/2006.06444, 2020. [Online]. Available: https://arxiv.org/abs/2006.06444

[2] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021. [Online]. Available: https://doi.org/10.1146/annurev-control-091420-084139

[3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022.

[4] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," *arXiv preprint arXiv:2201.07207*, 2022.

[5] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 335–346, 1990.

[6] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[7] X. Chen, X. Wang, S. Changpinyo, A. Piergiovanni, P. Padlewski, D. Salz, S. Goodman, A. Grycner, B. Mustafa, L. Beyer *et al.*, "Pali: A jointly-scaled multilingual language-image model," *arXiv preprint arXiv:2209.06794*, 2022.

[8] O. Shorinwa, T. Halsted, J. Yu, and M. Schwager, "Distributed Optimization Methods for Multi-Robot Systems: Part I — A Tutorial," 2023, under Review.

[9] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018. [Online]. Available: https://doi.org/10.1146/annurev-control-060117-105157

[10] D. P. Losey, M. Li, J. Bohg, and D. Sadigh, "Learning from my partner's actions: Roles in decentralized robot teams," in *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, 2019.

[11] S. Choudhury, J. Gupta, M. J. Kochenderfer, D. Sadigh, and J. Bohg, "Dynamic multi-robot task allocation under uncertainty and temporal constraints," 2022.

[12] H. Du, W. Zhu, G. Wen, Z. Duan, and J. Lü, "Distributed formation control of multiple quadrotor aircraft based on nonsmooth consensus algorithms," *IEEE transactions on cybernetics*, vol. 49, no. 1, pp. 342–353, 2017.

[13] S. Hayat, E. Yanmaz, C. Bettstetter, and T. X. Brown, "Multi-objective drone path planning for search and rescue with quality-of-service requirements," *Autonomous Robots*, vol. 44, no. 7, pp. 1183–1198, 2020.

[14] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *2013 IEEE International conference on robotics and automation*. IEEE, 2013, pp. 855–862.

[15] S. Nikolaidis, Y. X. Zhu, D. Hsu, and S. Srinivasa, "Human-robot mutual adaptation in shared autonomy," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 294–302.

[16] A. Xie, D. Losey, R. Tolsma, C. Finn, and D. Sadigh, "Learning latent representations to influence multi-agent interaction," in *Proceedings of the 4th Conference on Robot Learning (CoRL)*, 2020.

[17] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah, "Efficient model learning from joint-action demonstrations for human-robot collaborative tasks," in *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, 2015, pp. 189–196.

[18] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "Cobots: Robust symbiotic autonomous mobile service robots," in *Twenty-fourth international joint conference on artificial intelligence*. Citeseer, 2015.

[19] B. Hayes, D. Ullman, E. Alexander, C. Bank, and B. Scassellati, "People help robots who help others, not robots who help themselves," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2014, pp. 255–260.

[20] V. Srinivasan and L. Takayama, "Help me please: Robot politeness strategies for soliciting help from humans," in *Proc. of the 2016 CHI conference on human factors in computing systems*, 2016, pp. 4945–4955.

[21] S. Saunderson and G. Nejat, "Robots asking for favors: The effects of directness and familiarity on persuasive hri," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1793–1800, 2021.

[22] A. Nanavati, C. I. Mavrogiannis, K. Weatherwax, L. Takayama, M. Cakmak, and S. S. Srinivasa, "Modeling human helpfulness with individual and contextual factors for robot planning." in *Robotics: Science and Systems*, 2021.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[24] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-1: Robotics transformer for real-world control at scale," 2022. [Online]. Available: https://arxiv.org/abs/2212.06817

[25] T. Lew, S. Singh, M. Prats, J. Bingham, J. Weisz, B. Holson, X. Zhang, V. Sindhwani, Y. Lu, F. Xia, P. Xu, T. Zhang, J. Tan, and M. Gonzalez, "Robotic table wiping via reinforcement learning and whole-body trajectory optimization," 2022. [Online]. Available: https://arxiv.org/abs/2210.10865

[26] X. Xiao, T. Zhang, K. Choromanski, E. Lee, A. Francis, J. Varley, S. Tu, S. Singh, P. Xu, F. Xia *et al.*, "Learning model predictive controllers with real-time attention for real-world navigation," *arXiv preprint arXiv:2209.10780*, 2022.

[27] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022. [Online]. Available: https://arxiv.org/abs/2204.02311

[28] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[29] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[30] G. A. Miller, E. Galanter, and K. H. Pribram, *Plans and the structure of behavior*. Henry Holt and Co, 1960.

[31] R. A. Knepper, S. Tellex, A. Li, N. Roy, and D. Rus, "Recovering from failure by asking for help," *Autonomous Robots*, vol. 39, pp. 347–362, 2015.

[32] T. Zhu, P. Karlsson, and C. Bregler, "Simpose: Effectively learning densepose and surface normals of people from simulated data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*. Springer, 2020, pp. 225–242.

[33] R. C. Mittelhammer, G. G. Judge, and D. J. Miller, *Econometric Foundations*. USA: Cambridge University Press, 2000.

[34] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, V. Zhao, Y. Zhou, C.-C. Chang, I. Krivokon, W. Rusch, M. Pickett, P. Srinivasan, L. Man, K. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. Chi, and Q. Le, "Lamda: Language models for dialog applications," 2022. [Online]. Available: https://arxiv.org/abs/2201.08239

[35] OpenAI, "Introducing chatgpt," 2022. [Online]. Available: https://openai.com/blog/chatgpt