This paper appears in Biomimetic Intelligence and Robotics, 2026.
https://doi.org/10.1016/j.birob.2026.100277

Enhanced Reasoning and Task Planning for Surgical Autonomy Using Multi-Modal Large Language Models With Gradual Learning

Sadra Zargarzadeh[a], Jemima Okanlawon[a], Maryam Mirzaei[a], Mahan Mohammadi[b], Mahdi Tavakoli[c]

[a]*Department of Electrical and Computer Engineering, University of Alberta, 116 St & 85 Ave, Edmonton, T6G 2R3, Alberta, Canada*
[b]*Department of Computer Science, University of Toronto, 172 St. George St, Toronto, M5R 0A3, Ontario, Canada*
[c]*Department of Electrical and Computer Engineering, Department of Biomedical Engineering, University of Alberta, 116 St & 85 Ave, Edmonton, T6G 2R3, Alberta, Canada*

Abstract

Large Language Models (LLMs) have been widely adopted in robotic applications in recent years, but their ability in task planning of long-horizon and complex tasks remains a challenge. In this work, we present a gradual learning method to address this challenge and explore its usability in surgical training tasks that require high levels of reasoning, such as peg transfer and the sliding puzzle task. Experiments were conducted using the da Vinci Research Kit (dVRK), with environment feedback initiating follow-up prompts for the LLM when necessary, as well as in a simulation environment. Results showed that for complex tasks, the gradual learning method outperformed the direct approach in the LLM's task and motion planning, requiring fewer follow-up prompts and leading to higher success rates with faster execution. This suggests that for complex pseudo-surgical tasks, it is more efficient to have the LLM solve simpler versions of the task while incrementally increasing complexity, rather than tackling the full complex task at once. The approach shows promise for enhancing robot-assisted surgery where tasks are complex, long-horizon, and demand high-reasoning abilities.

Keywords: Large Language Models, Reasoning, Task Planning, Surgical Robotics, Embodied Learning for Robotics, Zero-shot Learning System for Robotics

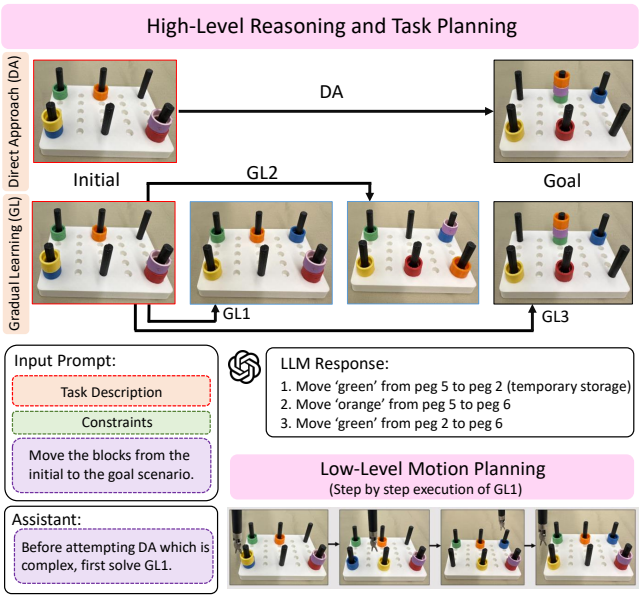

Figure 1: An example of Gradual Learning for task planning of a complex peg transfer task and its execution by the robot. Rather than attempting to solve the entire task at once, our method breaks down the task into simpler sub-problems, allowing the LLM to build on its reasoning capabilities incrementally. Full prompts are provided in the Appendix[4].

1. Introduction

The adoption of robots in surgery has changed the way many surgeons operate. Robot-assisted surgery (RAS), which is typically based on local or remote teleoperation with hand controllers or other input devices, allows the surgeon to perform tasks during an operation with greater precision, dexterity, and control [1]. While RAS has demonstrated clear advantages, its full reliance on human operators for critical decision-making and task planning has limited its potential for full autonomy. Surgical tasks, especially those requiring high levels of reasoning and 'long-horizon planning', present a significant challenge to autonomous systems due to the complexity, variability, and unpredictability of real-world operations. Adding elements of autonomy that would delegate support tasks to the robot to augment the dexterity of the surgeon has been a focus of research in recent years [2, 3, 4, 5, 6, 7].

Autonomy in surgical robots can be classified into six stages [8] listed below. Level 0: No autonomy, Level 1: robot assistance, Level 2: task autonomy, Level 3: conditional autonomy, Level 4: high autonomy, and Level 5: full automation. Achieving higher levels of autonomy in surgical robots could improve safety, reduce human error, and enable surgeons to focus on more critical aspects of surgery by offloading routine or repetitive tasks to the robotic system. The transition from level 3, where the operator selects and approves a surgical plan, after which the robot autonomously performs the procedure under close human supervision, to level 4, where the robot gains decision-making abilities but still operates under the oversight of a qualified operator, remains a significant challenge.

Autonomous surgical planning in robot-assisted surgical sub-tasks requires a reasoning system that mimics human decision-making, enabling preoperative task planning and real-time adjustments during surgery to address unexpected events. An important aspect of future human-robot teaming is the transparency of the robot's reasoning to the human user and vice versa, ensuring mutual understanding of decisions. This transparency improves collaboration, reduces risks, and enhances trust, as both parties can anticipate and adapt to each other's actions in dynamic environments.

The rise of Large Language Models (LLMs) and Vision Language Models (VLMs) has impacted robotics research in recent years [9, 10, 11]. LLMs have demonstrated remarkable capabilities in understanding and generating natural language, as well as solving complex reasoning problems in non-medical domains. These models can potentially be leveraged for high-level task planning in robotic systems, where they can reason about task sequences, adjust for unforeseen conditions, and generate executable plans for downstream motion control systems. Although the main focus of the integration has been on general object manipulation tasks [12, 13, 14, 15, 16, 17], it is slowly making its way into more specific domains such as robot-assisted surgery [18, 19, 20]. The reasoning capability of LLMs allows it to plan the task as the high-level reasoning unit, and pass this information to the mid-level motion planning and low-level motion control units for execution.

Surgical tasks are often complex and span long horizons, requiring detailed planning and adaptability over extended operations. For example, procedures like laparoscopic cholecystectomy or robotic-assisted suturing involve intricate, multi-step processes requiring precise planning, continuous decision-making, and adaptability to unexpected conditions like tissue deformation or bleeding. Long-horizon task planning remains an ongoing challenge

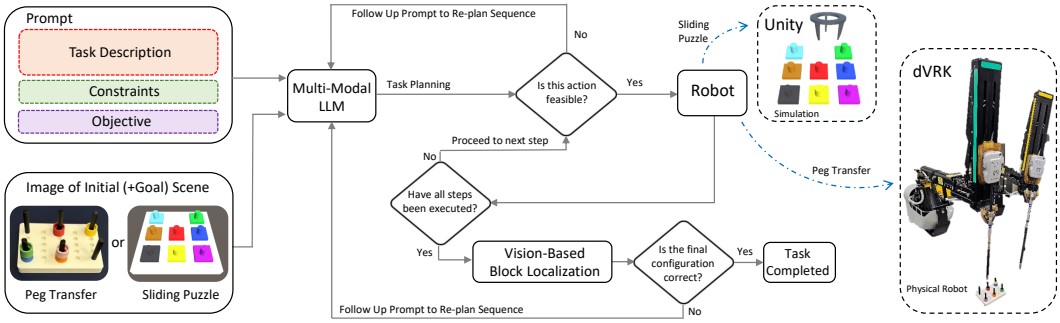

Figure 2: Schematic illustration of the overall system framework. The figure conceptually outlines the flow of task planning and execution, from the initial scene and task prompt to the multi-modal LLM, the robot platform (dVRK or Unity simulation), and the feedback loop for follow-up prompts.

in robotics, particularly in the medical domain.

To tackle this challenge in a more controlled environment, pseudo-surgical tasks like peg transfer and the sliding puzzle game are commonly used in robotic-assisted surgery training to replicate essential skills such as manual dexterity and cognitive reasoning. These tasks, while simpler than actual surgeries, introduce long-horizon planning challenges. For instance, transferring a single block becomes more complex when it involves multiple steps, similar to the multi-step planning required in real surgical procedures. This controlled setting allows for the focused development of autonomous systems capable of handling these complexities.

In this work, we propose a method called *Gradual Learning* to address this complexity in long-horizon task planning. Gradual learning is a step-wise method that enables LLMs to progressively solve complex reasoning-demanding tasks. Rather than attempting to solve the entire task at once, our proposed method breaks down the task into simpler sub-problems, allowing the LLM to build on its reasoning capabilities incrementally. This approach minimizes the complexity the LLM handles at a given time, preventing it from facing overly challenging tasks prematurely, and allows the robot to handle complex situations step by step, enhancing task planning capabilities.

Our contributions are as follows: 1) We propose a Gradual Learning framework for task planning in surgical robotics, leveraging the reasoning capabilities of LLMs and incorporating environment feedback to initiate follow-

up prompts, 2) we implement and evaluate the framework on two reasoning-demanding tasks, demonstrating its superiority over direct planning methods in terms of performance and task execution, and 3) we discuss the potential implications of gradual learning in advancing surgical autonomy and outline future directions for integrating LLMs into real-world surgical environments.

2. Related Work

2.1. Large Language Models for Robot Planning

Large Language Models (LLMs) have recently demonstrated significant potential in robot task and motion planning [21, 22, 23, 24, 15, 25, 26, 12, 27, 28, 29, 30, 31, 32]. Traditional methods rely on symbolic reasoning [33] and optimization [34] techniques, often requiring domain-specific models that limit generalization. LLM-based approaches, including our proposed gradual learning method, can reason and infer high-level plans through natural language, complementing motion planning units such as reinforcement learning by feeding them structured plans for execution. For instance, SayCan [35] leverages LLMs to generate task plans and assess action feasibility using affordance models. Other recent methods such as CurricuLLM [36], apply curriculum learning [37] by progressing from simpler to complex demonstrations in a static order. Similarly, our gradual learning method builds complexity incrementally but adds feedback to the LLM for dynamic re-planning, essential for robotic applications. These LLM-driven methods enhance both the flexibility and efficiency of robotic systems, optimizing task execution even in open-ended environments.

2.2. Existing Early Work on LLMs for Surgical Autonomy

Early work on LLMs in surgical autonomy has demonstrated the potential to enhance decision-making and task planning in robotic-assisted surgeries. These works have explored LLMs to facilitate reasoning and action generation in critical and dynamic environments where autonomous systems must adapt to unforeseen conditions. SUFIA [19] presents a framework that combines the reasoning power of LLMs with perception modules for augmented dexterity, enabling surgical robots to autonomously perform tasks like needle lifting and vessel dilation with natural language inputs. [38] introduces surgical action planning that generates future action plans from visual inputs to address the absence of intraoperative predictive planning. Similarly, our previous work on multi-modal LLMs for robot-assisted blood suction in

surgeries [18] demonstrates the ability of LLMs to prioritize and plan tasks, adapting to surgical complexities such as active bleeding and blood clots. These foundational approaches highlight the utility of LLMs in advancing the autonomy and safety of surgical robots, pushing the field toward more reliable and human-like robotic assistants.

2.3. Task Decomposition and Surgical Process Modeling

Traditional approaches to surgical autonomy have leveraged symbolic and hierarchical task representations to model complex procedures. Surgical Process Modeling (SPM) involves decomposing high-level procedures into phases, steps, and atomic actions, forming the basis for structured automation and decision support [39]. These models often draw from human demonstrations and annotations, creating deterministic state-transition systems that encode expert knowledge.

Recent efforts have applied formal planning languages such as the Planning Domain Definition Language (PDDL) to model surgical tasks with explicit preconditions and effects. For instance, [40] introduces a PDDL-based framework as a tutoring system for teaching surgical decision-making. Formal representations, such as finite-state task decomposition, have also been proposed. The COMPASS framework [41] introduces a unified formal model using motion primitives and context labels to structure surgical tasks, enabling multi-granularity task analysis and learning.

While these symbolic planners offer strong guarantees and interpretability, they typically require extensive manual modeling and are sensitive to variability in surgical scenes. In contrast, our proposed gradual learning framework offers an alternative that elicits similar task decomposition behavior from a general-purpose LLM using incremental prompting and feedback, without requiring explicit domain encoding or handcrafted transition models. This positions our method as a complementary paradigm that maintains structure and reasoning while improving adaptability and accessibility.

3. Task Description and Hierarchical Structure

The peg transfer and sliding puzzle tasks are well-suited for testing long-horizon task planning, as they simulate key aspects of surgical procedures. The peg transfer task mirrors surgical challenges like manipulating tissues or tools in confined spaces, while the sliding puzzle requires strategic planning and adaptability, similar to continuous decision-making in surgery. Both

tasks require precise movements and sequential actions, offering controlled settings to assess the gradual learning approach and the effectiveness of LLMs in improving task planning for robotic surgery.

3.1. Peg Transfer

The peg transfer task is a surgical training exercise focused on improving dexterity, hand-eye coordination, and ambidexterity by requiring trainees to switch hands. It's a key element of the Fundamentals of Laparoscopic Surgery (FLS) program [42], enhancing precision, bimanual coordination, and spatial awareness. We adapted the peg transfer task into a strategy game to test LLMs in logical reasoning and task planning. Our setup uses six pegs and six colored blocks, with constraints: each peg holds up to three blocks, only the top block can be moved, and blocks must be placed on top of others. The game's objective is to develop a step-by-step movement plan to complete a task or achieve a desired block configuration using the shortest possible path.

3.2. Sliding Puzzle

The sliding puzzle game requires planning and strategy, involving 8 blocks in a 3x3 grid with one empty space. Blocks are moved into the empty space until a goal configuration is achieved, with constraints such as no diagonal moves or direct swaps. The puzzle assesses task planning and logical reasoning, with tasks including moving a specific block to a goal position and rearranging the entire grid.

3.3. Hierarchical Structure

We follow a modular hierarchy consisting of three layers: (1) *High-Level Reasoning*, where a Large Language Model (LLM) constructs the task plans using the proposed Gradual Learning method; (2) *Mid-Level Motion Planning*, where a rule-based planner interprets symbolic plans and generates corresponding motion targets, such as gripper poses or block displacements, based on task-specific constraints; and (3) *Low-Level Motion Control*, where the physical robot dVRK and simulation environment executes these trajectories via motor updates.

The Gradual Learning module operates mainly within the high-level layer, producing context-informed symbolic actions that define the overall task structure. These actions are handed off to the mid-level motion plan module, while final execution is handled through the low-level control interface.

4. Methods

4.1. Gradual Learning with LLMs

In this work, we address the task planning problem $P = \langle S, A, T, s_{\text{init}}, G \rangle$ in an environment characterized by discrete, fully observable states, finite actions, and deterministic transitions. For each state $s \in S$, an action $a \in A$ can be chosen from the subset of applicable actions $A(s) \subseteq A$, provided that the necessary preconditions are met. The transition function $T : S \times A \to S$ defines the resulting state after an action is taken. The initial state is represented by $s_{\text{init}} \in S$, while $G \subseteq S$ is the set of goal states. A solution to this planning problem is a sequence of actions $\pi = (a_1, a_2, \ldots, a_n)$ that transforms the initial state s_{init} into one of the goal states. In long-horizon sequential task planning, the number of actions n typically grows quite large.

Algorithm 1: Gradual Learning for Task Planning

Input: Task planning problem $P = \langle S, A, T, s_{\text{init}}, G \rangle$
Output: Sequence of actions $\pi = (a_1, a_2, \ldots, a_n)$ leading to a goal state $s \in G$
Define simpler task planning problems $P_1, P_2, \ldots, P_k$ such that:
    P_1 is the simplest version of P
    P_k is the full complex task planning problem P
Initialize $i = 1$ {Start with the simplest problem P_1}
while $i \leq k$ **do**
    Select the current state $s \in S$ and set of applicable actions $A(s)$
    Prompt LLM with task planning problem P_i to find action $a_i \in A(s)$
    Use the transition function T to determine the next state $s' = T(s, a_i)$
    if $s' \in G$ **then**
        Increment i {Proceed to the next more complex task P_{i+1}}
    else
        Provide feedback and re-prompt with P_i
    end if
end while
Return: Solution $\pi = (a_1, a_2, \ldots, a_n)$ that transforms s_{init} into a goal state $s \in G$

We propose a method called *Gradual Learning*, where instead of attempting to solve a complex problem directly (*Direct Approach*), we solve a set of

simpler versions of the tasks and lead the way up for the LLM to learn in a gradual process as demonstrated in Algorithm 1. This approach reduces the complexity the LLM has to manage at any given time, preventing it from being overwhelmed by difficult tasks too early. It mirrors the way students learn increasingly challenging concepts as they advance through different grades in school, helping the model improve performance by tackling smaller, more manageable sub-problems first. We used the number of moves required to complete a task as an indicator of increasing complexity, guiding the transition from simple to complex tasks in our gradual learning approach.

To enhance transparency, we include representative prompt excerpts for both tasks in this section. For the peg transfer task, the LLM is instructed using a structured prompt that describes the initial and goal block configurations, defines spatial constraints and movement rules (e.g., "You can only move one block at a time", "Do not use temporary storage pegs unless absolutely necessary"), and emphasizes efficiency through principles such as minimizing physical work and reusing the closest peg for storage. For the sliding puzzle task, the prompt provides a grid-based representation of the 8-puzzle using matrix notation, with clear rules about allowed moves (e.g., "Only move a tile into an adjacent empty square horizontally or vertically") and a suggestion to use A* search for optimal planning. These examples reflect the actual prompts used in evaluation, with full prompt sequences included in the Appendix.

All experiments were conducted using GPT-4o, the multi-modal version released by OpenAI in May 2024, accessed via the public API. This model is a general-purpose large language model and was not fine-tuned on surgical data. It has no built-in domain knowledge beyond what is provided through the prompt (e.g., task rules, scene description, or feedback). This design choice emphasizes the accessibility of our method and highlights the robustness of the gradual learning framework in enabling structured reasoning and task planning without task-specific fine-tuning.

In both the gradual learning and direct approaches, the LLM is provided only with the game setup, constraints, and a set of guidance rules, but no problem examples or solutions, making all trials zero-shot. Feedback is given through 'follow-up prompts' whenever the LLM makes a mistake. Follow-up prompts are any prompts given after the initial task and solution attempt. They highlight the mistake and remind the LLM of the rules or task goal, or ask for an explanation of its strategy, without explicitly providing a solution. Testing showed that the LLM struggles to find mistakes independently

but, with guidance on the error's location, it can recognize and correct its mistakes, consistent with findings from [43]. Fig. 2 presents a schematic overview of the proposed system framework, illustrating the flow from the initial task prompt and scene input to the multi-modal LLM, which performs high-level reasoning to guide task planning. The generated plan is then executed on either a physical robot platform (dVRK) or within a Unity-based simulation environment, with a feedback loop enabling corrective follow-up prompts based on execution outcomes.

4.2. Peg Transfer Experimental Setup

For the peg transfer experiment, we 3D-printed a pegboard with vertically adjacent peg holes spaced 1 cm apart and horizontally adjacent holes spaced 2 cm apart. The pegboard was arranged in a Z-order with six pegs numbered 1 to 6 from top left to bottom right, with horizontally or vertically adjacent pegs spaced 4 cm apart.

The Patient Side Manipulator (PSM) robot of the da Vinci Research Kit (dVRK) [44] equipped with a large needle driver tool was used to perform the physical tasks. Execution of the motions relied on a Python function developed to move blocks between pegs. The function requires the peg coordinates, start and end pegs, and the number of blocks on the start peg as inputs, and determines the pickup height based on the number of blocks. Fig. 3 illustrates the robot's movement and end-effector trajectory when instructed to 'Move the green block from Peg 4 to Peg 3.'

The action space consists of discrete "pick-and-place" operations between pegs. An action is defined as `Move [color] from Peg [i] to Peg [j]`, where $i \neq j$ and both $i, j \in \{1, 2, ..., 6\}$. Additional constraints include: (1) only the topmost block of any peg can be picked up, and (2) no peg may contain more than 3 blocks after placement. These constraints were enforced via pre-checks and prompted back to the LLM in cases of violation.

The LLM occasionally generated motion plans that violated task constraints. Common errors included exceeding the peg capacity by attempting to place a fourth block on a peg, trying to move a block that was beneath another, incorrectly ordering block placement by suggesting a block could be placed underneath others, and using distant pegs for temporary storage when closer ones were available. Environment grounding was employed to automatically detect and address such issues. This allowed for real-time identification of violations, prompting the LLM with feedback to refine its plan.

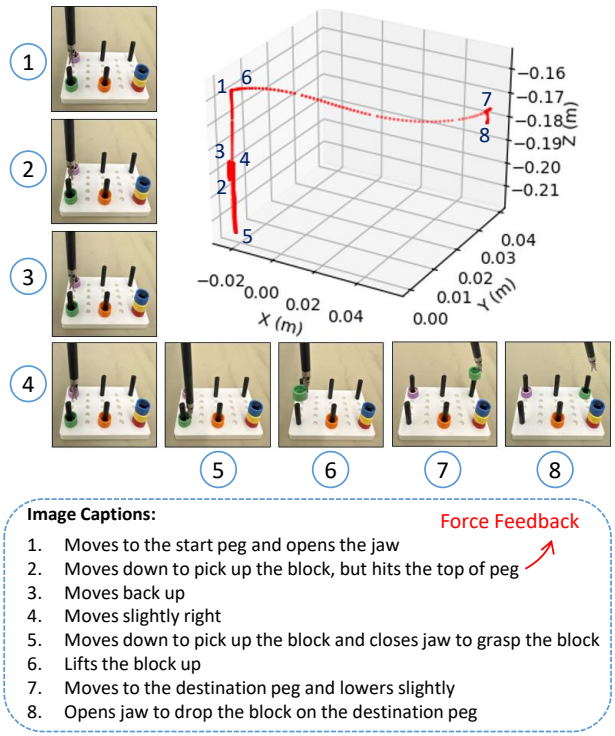

Figure 3: Robot movement and end-effector trajectory while executing the command: 'Move the green block from Peg 4 to Peg 3.'

The grounding process relied on two feedback modalities: image capture and force sensing, as illustrated in Fig. 3 and Fig. 4.

A webcam (Logitech International S.A., Lausanne, Switzerland) was used to visually analyze the pegboard layout, converting the captured images into text for comparison with the movement plan using computer vision techniques. A Python script, employing OpenCV, identified six black pegs and blocks in red, blue, green, purple, yellow, and orange. Blocks were grouped with their nearest peg based on x, and y coordinates within a small tolerance. This layout information was used to initialize the task for the LLM. If an error occurred, such as attempting to move a block beneath another, the system re-prompted the LLM with feedback such as, "Error in step 2. You cannot move a block that is underneath another block. Analyze and explain your mistake before retrying." The camera also verified the final block arrangement to prevent LLM hallucination errors. If discrepancies were found between the expected and actual configurations, the robot undid previous

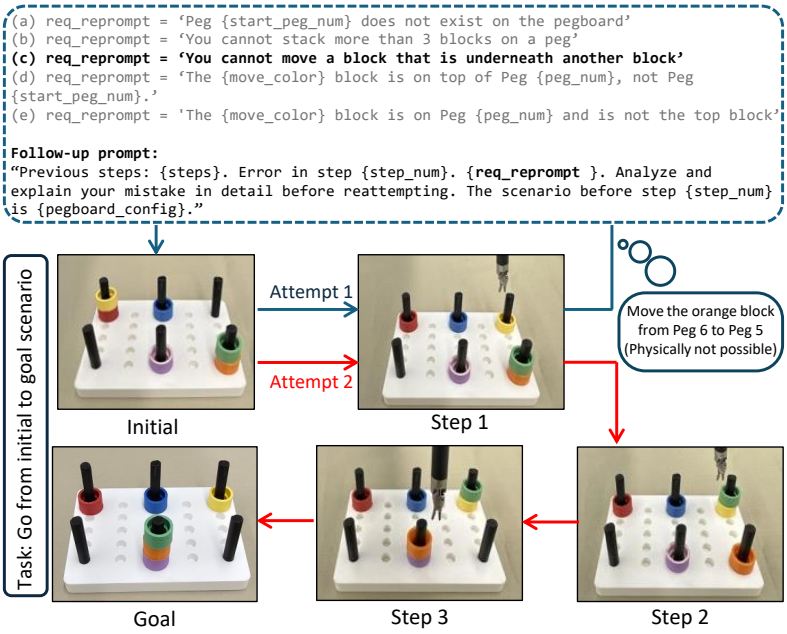

Figure 4: Grounded peg transfer experiment. The LLM's first attempt failed as it violated a constraint. The robot automatically undoes its previous steps whenever a correction prompt is required. The second attempt succeeded without violations, and the robot executed it correctly.

moves, and the LLM was re-prompted with details of the incorrect arrangement.

Force sensing was used to manage physical interactions between the robot and the pegboard. Feedback from the dVRK's actuators estimated forces on the end effector to account for pegboard shifts due to robot motion or disturbances. During block pickup, z-axis force was monitored, and if the robot encountered a peg, exceeding the force threshold, it adjusted its position before reattempting the move (Fig. 3). This force feedback influenced the robot's motion control but not the LLM's high-level plan.

4.3. Sliding Puzzle Experimental Setup

In the sliding puzzle experiment, we developed a simulated environment in Unity featuring eight colored blocks on a 3x3 grid, with one empty space to allow the blocks to move. For both simple and complex tasks, puzzle difficulty was defined by the number of optimal moves required to solve them. Both tasks involve puzzles requiring 1 to 10 moves to solve. The LLM's

performance was evaluated under two conditions: (1) with explicit guidance to use the A* algorithm and Manhattan distance heuristic, and (2) without such guidance.

The action space is limited to adjacent horizontal or vertical movements of blocks into the single empty grid cell. An action is represented as `Move [block_id] from (x1, y1) to (x2, y2)`, where (x_2, y_2) is the empty cell and the movement must satisfy Manhattan distance = 1. Diagonal or non-adjacent movements are invalid. These rules were provided as part of the initial task prompt and reinforced during follow-up re-prompts in case of violations.

The LLM generated its solutions step-by-step, and follow-up prompts were sent until the goal was reached or an error occurred. Transmission Control Protocol (TCP) was used to communicate between the LLM and the simulation environment. The processed outputs from the TCP connection were used as commands to move the colored blocks, which were rigid bodies in the Unity simulation.

The LLM's task planning is classified as optimal, non-optimal, or unsuccessful. Unsuccessful responses include invalid moves (e.g., diagonal, jumps), collisions, goal misidentification, or failure to solve the puzzle within 5 extra moves beyond the optimal number.

5. Results

5.1. Peg Transfer

5.1.1. Gradual Learning vs Direct Approach

The gradual learning method was compared with the direct approach for two distinct peg transfer tasks: a simple task and a complex task. In the simple task, the goal was to move a single block to a target peg. For instance, in Level 1, the task involved moving the top block to a given peg using a specific pegboard orientation. In Level 2, the middle block (second from the top) was moved, while Level 3 required moving the bottom block (third from the top). Although the exact moves were not transferable across levels due to the changing scenarios, the underlying concepts remained consistent.

In contrast, the complex task required reconfiguring a set of blocks from an inital scenario to a goal scenario. Here, Levels 1-3 corresponded to progressively building toward the full solution required in Level 4. Specifically, Level 1 involved one move, Level 2 required two moves, Level 3 needed three moves, and Level 4 demanded four to five moves.

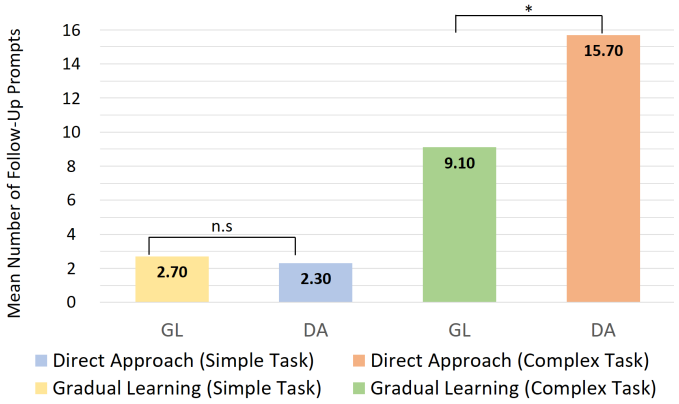

Figure 5: Average number of follow-up prompts required to complete the peg transfer task across different difficulty levels. Results compare the GL method against the DA for both simple and complex task variants. Lower values indicate better performance. Statistical significance was determined using paired t-tests.

As shown in Fig. 5 and detailed in Table I, the gradual learning approach required about 42% fewer follow-up prompts on average to solve levels 1-4 compared to solving level 4 alone using the direct approach for the complex task, with a p-value<0.05 from a paired t-test. However, for the simpler task, there was no statistically significant difference between the two approaches. Additionally, the total time for LLM processing and robot execution, including necessary follow-ups and corrections, was reduced by 23.8%, decreasing from 596 seconds in the direct approach to 454 seconds with the gradual learning method. This indicates that gradual learning is more effective for complex tasks that require long-horizon planning.

Qualitative observations showed that while gradual learning did not stop the LLM from repeating mistakes in more challenging levels, it enabled quicker error detection and correction. In some cases, a certain block did not need to be moved when transitioning to the goal scenario, but the LLM incorrectly stated to move the block. With gradual learning, the LLM quickly recognized and fixed this mistake in each level. In contrast, with the direct approach, it alternated between correcting the same two errors, fixing one only to repeat the other.

Table 1: Number of follow-up prompts required in gradual learning (GL) and direct approach (DA) for the simple and complex peg transfer tasks.

	Trials	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Simple Task	Level 1 (GL)	0	0	1	1	0	0	0	1	0	1
	Level 2 (GL)	3	0	1	3	2	2	5	1	1	1
	Level 3 (GL)	2	3	4	3	1	3	2	9	6	8
	GL Total	**5**	**3**	**6**	**7**	**3**	**5**	**7**	**11**	**7**	**10**
	Level 3 (DA)	**3**	**7**	**4**	**2**	**3**	**4**	**3**	**3**	**7**	**3**
Complex Task	Level 1 (GL)	1	0	0	0	0	0	0	0	2	0
	Level 2 (GL)	0	0	1	0	2	2	3	0	0	1
	Level 3 (GL)	2	0	4	2	4	1	5	8	1	2
	Level 4 (GL)	0	0	6	13	3	11	3	9	0	5
	GL Total	**3**	**0**	**11**	**15**	**9**	**14**	**11**	**17**	**3**	**8**
	Level 4 (DA)	**6**	**10**	**17**	**22**	**11**	**32**	**26**	**7**	**2**	**24**

5.2. Sliding Puzzle

5.2.1. Gradual Learning vs Direct Approach

The sliding puzzle task was tested using both the gradual learning approach and the direct approach to compare their effectiveness in solving the sliding puzzles. Two different tasks were designed for evaluation. Task 1 required moving a specific block to a goal position and Task 2 required moving all blocks from an initial configuration to a goal configuration. The LLM's task planning was categorized as optimal, non-optimal, or unsuccessful, depending on whether it solved the puzzle in the minimum number of moves (OPT), exceeded the optimal number of moves (NOPT), or failed to solve the puzzle due to invalid moves (1MR and $\geq$ 2MR).

The complexity of the puzzles varied, requiring between 1 and 10 moves to solve with each puzzle being repeated for 10 trials. The 1-move to 5-move puzzles were solvable using all approaches, and the success rates (normalized with the 10 trials) of the 6-move to 10-move puzzles are shown in Fig. 6. As seen in Fig. 6, in task 1, the gradual learning method achieved a perfect success rate in all trials, scoring 5 (4.8 optimal), while the direct approach scored 4.5 (4 optimal). In task 2, gradual learning also led to a higher success rate of 4.4 compared to 3.3 with the direct approach. The LLM's success rate improved with guidance on the A* algorithm, emphasizing the importance of

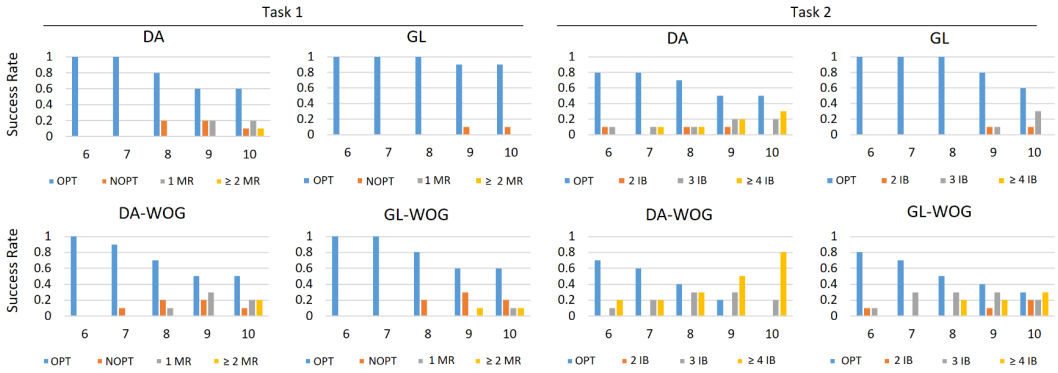

Figure 6: Sliding puzzle task success rates across increasing difficulty levels (6-move to 10-move puzzles), normalized over 10 trials per condition. Task outcomes are classified as OPT (optimal completion), NOPT (non-optimal but successful), 1MR (one move remained), and $\geq$ 2MR (two or more moves remained). The GL method outperforms the DA in both overall success rate and distribution of optimal completions.

structured prompts and contextual cues, and consistent with findings from previous work [45].

Qualitative observations highlighted that the gradual learning method enabled the LLM to recognize and correct mistakes more effectively than the direct approach. However, similar to the peg transfer task, the LLM sometimes repeated the same errors at higher difficulty levels, even after having resolved them in simpler stages. For instance, in some trials, the model misidentified the goal state and executed incorrect moves. Despite this, the gradual learning framework allowed the LLM to recover more quickly by leveraging intermediate feedback and structured prompts.

In contrast, the direct approach frequently struggled with goal misidentification, particularly in Task 2. The LLM often terminated planning prematurely after reaching an intermediate configuration that visually resembled, but did not match, the goal. Such errors were not easily rectified, even with follow-up prompting. Another recurring issue was the generation of invalid actions, such as diagonal moves or illegal block swaps, which violated task constraints. These faults, when uncorrected early, often caused cascading failures throughout the plan.

The gradual learning method, by contrast, facilitated earlier detection and recovery from such mistakes. In multi-step puzzles, the presence of well-defined sub-problems helped constrain the LLM's search space and reduced

task ambiguity. This intermediate anchoring led to more stable reasoning across planning steps. Moreover, when the model made an invalid move during a subtask (e.g., attempting to swap two blocks in Level 2 of Task 1), a clarification prompt typically led to successful self-correction in the next attempt.

6. Discussion and Limitations

The results from both tasks, the peg transfer and sliding puzzle, consistently highlight the effectiveness of the Gradual Learning (GL) approach compared to the Direct Approach (DA). In both tasks, the GL method reduced errors, follow-up prompts, and overall task completion time. This demonstrates that incrementally increasing task complexity allows the LLM to better manage the complex load and improve task performance, making it an appropriate method for long-horizon, multi-step planning in robotic systems. However, the nature of errors in each task provides valuable insights. In the peg transfer task, errors stemmed primarily from physical constraint violations, such as attempting to move blocks that were buried, underscoring the importance of dexterity and spatial awareness. In contrast, the sliding puzzle task involved logical reasoning and spatial planning, with errors arising from miscalculating move sequences or misidentifying goal states. These differences suggest that while the GL approach is broadly applicable, the type of feedback needed, whether physical or logical, may vary depending on the task. Overall, the adaptability of GL across these diverse environments suggests its potential for enhancing performance not just in robotic surgery, but also across a wide range of robotic tasks.

While this study shows promising results, there are limitations and opportunities for improvement. The feedback mechanisms can be generalized to handle more error types, enhancing adaptability to real-world scenarios like surgery, where tissue deformation or bleeding frequently arises. Additionally, while the current focus has been on deterministic tasks, the GL approach can be expanded to thrive in stochastic environments, allowing it to handle unexpected events with more robust task planning. Moreover, a deeper study is needed to generalize task breakdowns from simple to complex, essential for improving this approach's flexibility and scalability.

7. Conclusion and Future Work

We proposed gradual learning, a method for long-horizon planning using LLMs, tested on the dVRK physical robot for the peg transfer and in simulation for the sliding puzzle tasks. Results showed that gradual learning outperforms the direct approach method, requiring fewer follow-up prompts and achieving faster execution with higher success rates. This method holds the potential for complex, reasoning-intensive tasks in robot-assisted surgery.

In future work, we aim to broaden the experimental scope by incorporating additional surgical training tasks with varied spatial and temporal dependencies, such as knot tying or multi-arm coordination. These domains would introduce new forms of uncertainty and challenge the framework's ability to reason across temporally extended actions and tool interactions. Furthermore, expanding the number of test scenarios, including out-of-distribution cases or multi-agent collaboration settings, could provide a more comprehensive validation of the framework's robustness. Finally, benchmarking against additional LLM prompting strategies and structured reasoning baselines would further contextualize the effectiveness of gradual learning in high-stakes, high-reasoning environments.

Future work will also focus on applying this method to tasks such as suturing with real-time feedback and exploring how the approach can be extended to stochastic environments, commonly encountered in real surgeries, where dynamic and unpredictable factors, such as tissue deformation or sudden bleeding, require more adaptive and robust task planning mechanisms.

8. Acknowledgements

This research was supported by the Canada Foundation for Innovation (CFI), the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Canadian Institutes of Health Research (CIHR), and Alberta Innovates.

9. Appendix

9.1. Peg Transfer Prompt

There is a pegboard containing 6 pegs arranged in 2 rows of 3. The pegs are numbered 1 to 3 in the top row (left to right) and pegs 4 to 6 in the bottom row (left to right). You are a very smart puzzle and math problem-solving human. You will use logic and reasoning to solve hard problems in

the simplest way. You always follow all the rules precisely and fully. It is critical that you calculate and do the least amount of physical work possible for every task; otherwise, you will lose energy.

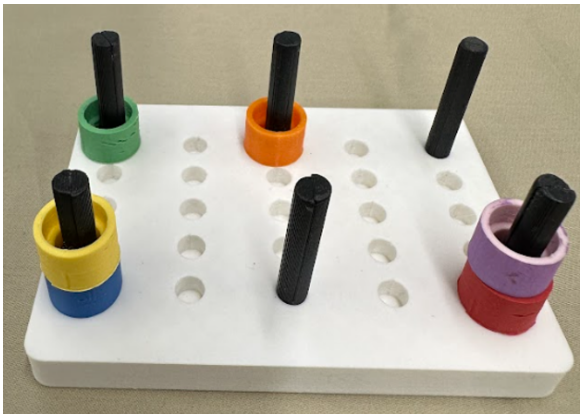

Figure 7: Peg Transfer - Initial Scenario

Initial Scenario: Pegs 3 and 5 have no blocks. On Peg 1, there is a green block, on Peg 2, there is an orange block, on Peg 4, there are two blocks (yellow on top of blue), and on Peg 6, there are two blocks (pink on top of red).

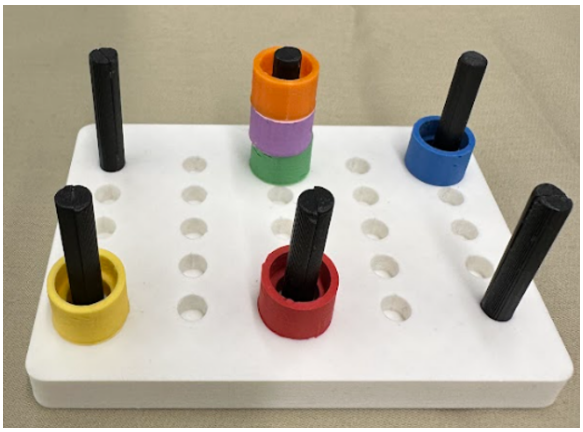

Figure 8: Peg Transfer - Goal Scenario

Goal Scenario: Pegs 1 and 6 have no blocks. On Peg 2, there are three blocks (orange on top of pink on top of green), on Peg 3, there is a blue block, on Peg 4, there is a yellow block, and on Peg 5, there is a red block.

Pegboard orientation 1: Peg 1 is at (1, 2). Peg 2 is at (2, 2). Peg 3 is at (3, 2). Peg 4 is at (1, 1). Peg 5 is at (2, 1). Peg 6 is at (3, 1).
Constraint A: Each peg can hold any number of blocks, provided the total number of blocks on the peg does not exceed 3.
Constraint B: You cannot move a block that is underneath another block.
Constraint C: Blocks are always placed on top of the topmost block on a peg.
Constraint D: You can only move one block at a time.
Constraint E: It is not possible for the same block to be on more than one peg at a time.
Rule 1: Use the shortest path possible for every task.
Rule 2: Always check the number of blocks on each peg before every step.
Rule 3: Always check if another block is on top of the block you want to move.
Rule 4: The top blocks can be moved with only one step.
Rule 5: Do not consider emptiness when selecting pegs.
Rule 6: Always consider every peg as an option unless it directly violates a constraint.
Rule 7: Before selecting a peg for temporary storage, calculate the distances to all pegs.
Rule 8: Prefer reusing the absolute closest peg for temporary storage of multiple blocks if it minimizes the total travel distance.
Rule 9: Assume that the starting position is in the geometric center of the arrangement of all pegs.
Rule 10: Do not move a block to the same peg that it's already on.
Rule 11: Minimize the physical work for every task.
Rule 12: Always try using the absolute shortest path, then if you run into issues start again.
Rule 13: You do not need to move cleared blocks back to their initial position.
Rule 14: You are allowed to use the target peg for clearing.
Rule 15: Never assume a peg becomes occupied after you place a block on it. A peg is not occupied unless it has at least 3 blocks on it.
Rule 16: Optimize only the task at hand. Do not overthink it.
Rule 17: Pay attention to the order of blocks on each peg and how the order of moves impacts the result.
Rule 18: Do not use temporary storage pegs unless absolutely necessary.

Objective: According to the instructions and rules of the puzzle game mentioned below, move the blocks from the initial scenario to the goal scenario.

9.2. Sliding Puzzle Prompt

I want you to help me solve a puzzle example of the 8-puzzle game. In the 8-puzzle game, there are 8 square blocks and an empty square. We will represent a puzzle configuration for the 8-puzzle game as a Python Numpy array (matrix) and image where the square blocks will each be designated with a number between 1 and 8 and the empty square will be designated as the number 0. So, we would like to start from an initial puzzle configuration and reach a final puzzle configuration. (1: orange, 2: green, 3: yellow, 4: red, 5: blue, 6: purple, 7: cyan, 8: black)

We would like to go from the following initial puzzle configuration to the goal configuration.

```
[[2, 1, 6],
[7, 5, 8],
[0, 4, 3]]
```

Figure 9: Sliding Puzzle - Initial Configuration

```
[[2, 6, 4],
[5, 1, 0],
[7, 3, 8]]
```

Figure 10: Sliding Puzzle - Goal Configuration

Instruction and rules:

Based on what is already known about the 8-puzzle game, here is some information you need to know about the rules and instructions of this puzzle game: In this puzzle game, in each step, a single move of a block must be made. So, at each step, a block can be moved by only moving into the adjacent empty square (0)

vertically or horizontally. More precisely, at each step, a block at position (i, j) in the matrix can be moved in one of the following ways:

The block can move upward to the adjacent position (i, j+1) if the adjacent empty square (0) is above it (at position (i, j+1)). In this case, just swap the position of the block with that of the empty square (0).
The block can move downward to the adjacent position (i, j-1) if the adjacent empty square (0) is below it (at position (i, j-1)). In this case, just swap the position of the block with that of the empty square (0).
The block can move leftward to the adjacent position (i-1, j) if the adjacent empty square (0) is to the left of it (at position (i-1, j)). In this case, just swap the position of the block with that of the empty square (0).
The block can move rightward to the adjacent position (i+1, j) if the adjacent empty square (0) is to the right of it (at position (i+1, j)). In this case, just swap the position of the block with that of the empty square (0).
Overall, in each step, only a single block must be moved, and that block can swap its position with that of the empty square (0) adjacent to it either on the left, right, below, or above.
In every time step, diagonal movement is not allowed. In other words, within a time step, we cannot make a move that changes both the i-component and j-component of the block we want to move. Do not make a diagonal move and no jump move is allowed. Only vertical or horizontal move to the adjacent empty square (0) is allowed.
(if with guidance) In order to choose which of the blocks adjacent to the empty square (0) to move, use the A* search algorithm so that the goal is reached in as few moves as possible. If there are several choices of move to pick, use the A* search algorithm and consider what happens after each choice is made, and then, pick one that yields the most efficient outcome.
In each step, mention and keep track of the position of each block as well as the position of the empty square. Based on everything mentioned above, let's optimally solve the puzzle step by step.

Objective: According to the instructions and rules of the puzzle game mentioned below, move the blocks from the initial puzzle configuration to the goal puzzle configuration.

References

[1] R. H. Taylor, A. Menciassi, G. Fichtinger, P. Fiorini, P. Dario, Medical robotics and computer-integrated surgery, Springer handbook of

robotics (2016) 1657–1684.

[2] P. Fiorini, K. Y. Goldberg, Y. Liu, R. H. Taylor, Concepts and trends in autonomy for robot-assisted surgery, Proceedings of the IEEE 110 (7) (2022) 993–1011.

[3] Y. Ou, S. Zargarzadeh, M. Tavakoli, Robot learning incorporating human interventions in the real world for autonomous surgical endoscopic camera control., J. Medical Robotics Res. 8 (3&4) (2023) 2340004–1.

[4] A. Pore, D. Corsi, E. Marchesini, D. Dall'Alba, A. Casals, A. Farinelli, P. Fiorini, Safe reinforcement learning using formal verification for tissue retraction in autonomous robotic-assisted surgery, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 4025–4031.

[5] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen, K. Goldberg, Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2016, pp. 4178–4185.

[6] Y. Ou, A. Soleymani, X. Li, M. Tavakoli, Autonomous blood suction for robot-assisted surgery: A sim-to-real reinforcement learning approach, IEEE Robotics and Automation Letters 9 (8) (2024) 7246–7253.

[7] Y. Ou, S. Zargarzadeh, P. Sedighi, M. Tavakoli, A realistic surgical simulator for non-rigid and contact-rich manipulation in surgeries with the da vinci research kit, in: 2024 21st International Conference on Ubiquitous Robots (UR), IEEE, 2024, pp. 64–70.

[8] G.-Z. Yang, J. Cambias, K. Cleary, E. Daimler, J. Drake, P. E. Dupont, N. Hata, P. Kazanzides, S. Martel, R. V. Patel, et al., Medical robotics—regulatory, ethical, and legal considerations for increasing levels of autonomy (2017).

[9] Y. Kim, D. Kim, J. Choi, J. Park, N. Oh, D. Park, A survey on integration of large language models with intelligent robots, Intelligent Service Robotics 17 (5) (2024) 1091–1107.

[10] J. Wang, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang, Y. Yao, X. Liu, B. Ge, S. Zhang, Large language models for robotics: Opportunities, challenges, and perspectives, Journal of Automation and Intelligence 4 (1) (2025) 52–64.

[11] C. Zhang, J. Chen, J. Li, Y. Peng, Z. Mao, Large language models for human-robot interaction: A review, Biomimetic Intelligence and Robotics (2023) 100131.

[12] S. S. Kannan, V. L. Venkatesh, B.-C. Min, Smart-llm: Smart multi-agent robot task planning using large language models, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2024, pp. 12140–12147.

[13] H. Liu, Y. Zhu, K. Kato, A. Tsukahara, I. Kondo, T. Aoyama, Y. Hasegawa, Enhancing the llm-based robot manipulation through human-robot collaboration, IEEE Robotics and Automation Letters 9 (8) (2024) 6904–6911.

[14] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, T. Funkhouser, Tidybot: Personalized robot assistance with large language models, Autonomous Robots 47 (8) (2023) 1087–1102.

[15] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, A. Garg, Progprompt: Generating situated robot task plans using large language models, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2023, pp. 11523–11530.

[16] Z. Mandi, S. Jain, S. Song, Roco: Dialectic multi-robot collaboration with large language models, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 286–299.

[17] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, D. Sadigh, Physically grounded vision-language models for robotic manipulation, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 12462–12469.

[18] S. Zargarzadeh, M. Mirzaei, Y. Ou, M. Tavakoli, From decision to action in surgical autonomy: Multi-modal large language models for robot-assisted blood suction, IEEE Robotics and Automation Letters (2025).

[19] M. Moghani, L. Doorenbos, W. C.-H. Panitch, S. Huver, M. Azizian, K. Goldberg, A. Garg, Sufia: language-guided augmented dexterity for robotic surgical assistants, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2024, pp. 6969–6976.

[20] G. Wang, L. Bai, W. J. Nah, J. Wang, Z. Zhang, Z. Chen, J. Wu, M. Islam, H. Liu, H. Ren, Surgical-lvlm: Learning to adapt large vision-language model for grounded visual question answering in robotic surgery, in: ICLR 2025 Workshop on Foundation Models in the Wild.

[21] Y. Ding, X. Zhang, C. Paxton, S. Zhang, Task and motion planning with large language models for object rearrangement, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2023, pp. 2086–2092.

[22] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, C. Fan, Autotamp: Autoregressive task and motion planning with llms as translators and checkers, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 6695–6702.

[23] Z. Zhao, W. S. Lee, D. Hsu, Large language models as commonsense knowledge for large-scale task planning, Advances in Neural Information Processing Systems 36 (2024).

[24] L. Sun, D. K. Jha, C. Hori, S. Jain, R. Corcodel, X. Zhu, M. Tomizuka, D. Romeres, Interactive planning using large language models for partially observable robotic tasks, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 14054–14061.

[25] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, H. Liu, Llmˆ 3: Large language model-based task and motion planning with motion failure reasoning, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2024, pp. 12086–12092.

[26] Y. Ding, X. Zhang, C. Paxton, S. Zhang, Leveraging commonsense knowledge from large language models for task and motion planning,

in: RSS 2023 Workshop on Learning for Task and Motion Planning, 2023.

[27] Z. Zhou, J. Song, K. Yao, Z. Shu, L. Ma, Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 2081–2088.

[28] R. Arora, S. Singh, K. Swaminathan, A. Datta, S. Banerjee, B. Bhowmick, K. M. Jatavallabhula, M. Sridharan, M. Krishna, Anticipate & act: Integrating llms and classical planning for efficient task execution in household environments, in: International Conference on Robotics and Automation, 2024.

[29] Z. Long, G. Killick, R. McCreadie, G. Aragon-Camarasa, Robollm: Robotic vision tasks grounded on multimodal large language models, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 12428–12435.

[30] S. Izquierdo-Badiola, G. Canal, C. Rizzo, G. Alenyà, Plancollabnl: leveraging large language models for adaptive plan generation in human-robot collaboration, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2024, pp. 17344–17350.

[31] 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, in: Conference on Robot Learning, PMLR, 2023, pp. 1769–1782.

[32] W. Huang, P. Abbeel, D. Pathak, I. Mordatch, Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, in: International conference on machine learning, PMLR, 2022, pp. 9118–9147.

[33] M. Sridharan, M. Gelfond, S. Zhang, J. Wyatt, Reba: A refinement-based architecture for knowledge representation and reasoning in robotics, Journal of Artificial Intelligence Research 65 (2019) 87–180.

[34] T. Marcucci, M. Petersen, D. von Wrangel, R. Tedrake, Motion planning around obstacles with convex optimization, Science robotics 8 (84) (2023) eadf7843.

[35] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al., Do as i can, not as i say: Grounding language in robotic affordances, in: Conference on robot learning, PMLR, 2023, pp. 287–318.

[36] K. Ryu, Q. Liao, Z. Li, P. Delgosha, K. Sreenath, N. Mehr, Curricullm: Automatic task curricula design for learning complex robot skills using large language models, in: 2025 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2025, pp. 4470–4477.

[37] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th annual international conference on machine learning, 2009, pp. 41–48.

[38] M. Xu, Z. Huang, J. Zhang, X. Zhang, Q. Dou, Surgical action planning with large language models, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2025, pp. 563–572.

[39] F. Lalys, P. Jannin, Surgical process modelling: a review, International Journal of Computer Assisted Radiology and Surgery 9 (2014) 495–511.

[40] N. Vannaprathip, P. Haddawy, H. Schultheis, S. Suebnukarn, Intelligent tutoring for surgical decision making: a planning-based approach, International Journal of Artificial Intelligence in Education 32 (2) (2022) 350–381.

[41] K. Hutchinson, I. Reyes, Z. Li, H. Alemzadeh, Compass: a formal framework and aggregate dataset for generalized surgical procedure modeling, International Journal of Computer Assisted Radiology and Surgery 18 (12) (2023) 2143–2154.

[42] G. Sroka, L. S. Feldman, M. C. Vassiliou, P. A. Kaneva, R. Fayez, G. M. Fried, Fundamentals of laparoscopic surgery simulator training to proficiency improves laparoscopic performance in the operating room—a randomized controlled trial, The American journal of surgery 199 (1) (2010) 115–120.

[43] G. Tyen, H. Mansoor, V. Cărbune, Y. P. Chen, T. Mak, Llms cannot find reasoning errors, but can correct them given the error location, in:

Findings of the Association for Computational Linguistics: ACL 2024, 2024, pp. 13894–13908.

[44] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, S. P. DiMaio, An open-source research kit for the da vinci® surgical system, in: 2014 IEEE international conference on robotics and automation (ICRA), IEEE, 2014, pp. 6434–6439.

[45] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, et al., Graph of thoughts: Solving elaborate problems with large language models, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 38, 2024, pp. 17682–17690.