Enhancing Robotic Manipulation with Large Language Models and Teleoperation: A Fault-Tolerant Approach to Uncertain Environments

Jiankun Wei, Dr. Lueder Kahrs
The Medical Computer Vision and Robotics Lab
Department of Computer Science
University of Toronto
Toronto, ON

Abstract

This paper presents an innovative approach to robotic manipulation, addressing limitations inherent in conventional robot motion planning algorithms such as MoveIt[7]. Traditionally, these algorithms require precise destination coordinates to generate movement sequences, limiting their utility in dynamic, uncertain environments. Furthermore, the lack of fault tolerance in these sequences compounds the challenge, as any error in execution, such as a failed object pickup, leads to the continuation of the sequence without corrective action. To overcome these obstacles, this project integrates Large Language Models (LLMs) with teleoperation techniques, focusing on a task that involves a da Vinci Research Kit (dVRK) endowrist[3] maneuvering a red cube towards a green cube without prior knowledge of exact object locations. The task abstraction simplifies the dVRK endowrist to a slim cylindrical bar, interacting solely with the bar to accomplish the task based on visual inputs from a Unity-based simulated environment. This approach leverages the Google Gemini model[1] to iteratively determine the bar's optimal positioning and orientation, enabling the system to adaptively and iteratively navigate toward the goal with enhanced fault tolerance and flexibility.

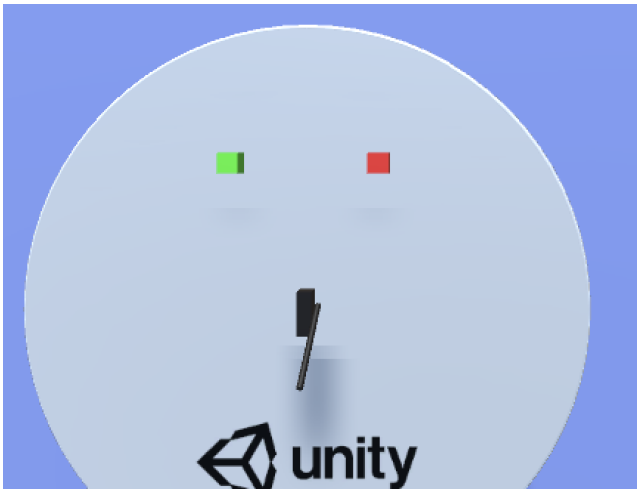


Figure 1: Unity Setup

Contents

1 Introduction

Robotic manipulation in uncertain and dynamic environments poses significant challenges for traditional motion planning algorithms. These algorithms, exemplified by MoveIt, excel in scenarios where destination coordinates are precisely known. However, real-world applications often lack this level of certainty, requiring a more adaptable and resilient approach to robotic control. Moreover, the rigidity of pre-determined motion sequences in these algorithms leaves little room for error correction, a critical flaw when interacting with unpredictable elements or in tasks requiring high precision.

In response to these challenges, this work introduces a novel integration of Large Language Models (LLMs) with teleoperation techniques, pushing the boundaries of robotic manipulation. By abstracting complex robotic systems into more manageable representations and leveraging the adaptive reasoning capabilities of LLMs, this paper propose a solution that not only addresses the issue of uncertain target locations but also introduces an inherent fault tolerance mechanism.

The experimental setup[9] involves a simulated task where a dVRK endowrist, represented as a simple cylindrical bar, is tasked by pushing a red cube toward a green cube. (See Figure 1) The complexity of this task is compounded by the lack of precise location data for the cubes, requiring the system to infer spatial relationships and appropriate actions solely from visual inputs provided by a simulated Unity environment. The core of our approach lies in the iterative consultation of the Google Gemini LLM, which guides the teleoperated movement of the bar toward the successful completion of the task.

This paper details the methodology behind this integration, the design of the experimental task, and the implications of our findings for the future of robotic manipulation in uncertain environments. By bridging the gap between the precision of traditional motion planning algorithms and the adaptability required for real-world applications, this research paves the way for more versatile and resilient robotic systems.

2 Related Works

The integration of Large Language Models (LLMs) in robotics has emerged as a significant area of research, particularly for tasks requiring a combination of common sense reasoning, environmental awareness, and logical deduction over extended periods or 'long-horizon' tasks. In this context, several notable works have contributed to the development of frameworks and algorithms that harness the cognitive capabilities of LLMs for robotic manipulation and planning.

One of the pioneering approaches in this domain is the Task2Motion model[4] introduced by Kevin Lin and colleagues. This model represents an innovative application of LLMs for robotic tasks, specifically in the context of picking and placing objects. Lin et al. successfully bridged the gap between long-horizon planning and real-time motion execution by employing a hybrid algorithm that amalgamates the 'shooting' method which computes the entire action sequence at once, and a 'greedy search' strategy that formulates movements incrementally. Their method demonstrated impressive proficiency in navigating around obstacles and achieving objectives efficiently.

Furthering the intersection of LLMs with robotic control, the LLM-Planner[8] developed by Chan Hee Song et al. capitalizes on the combination of LLMs' strategic reasoning and pre-trained robotic primitives' operational capabilities. While recognizing the adept reasoning of LLMs, Song et al. acknowledged their limitations in interfacing directly with robot hardware. Conversely, robotic primitives excel in control tasks but lack overarching strategic guidance. The LLM-Planner addresses this by using LLM to decompose complex tasks into executable sub-tasks, then calling low-level planners to align short-term primitive actions with long-term goals, thereby enhancing task completion efficacy.

Additionally, the Language to Reward model[2] by Google DeepMind explores the utility of LLMs in a different facet—reward function construction for reinforcement learning (RL) in robotics. By

prompting LLMs to generate parameterized code, they crafted reward functions that facilitate the training of RL algorithms, thereby smoothing the path for intricate robotic movements.

Recent advancements, such as those presented in the publications of ChatGPT-4[5] and Gemini Pro Vision, have expanded LLM functionalities beyond textual processing, introducing capabilities to interpret visual inputs. This groundbreaking progress enables LLMs to infer spatial relationships from images, a development that this paper capitalizes on. By feeding with visual data, LLMs extract critical positioning information, empowering the robotic system to make accurate orientation decisions—thereby highlighting the potential for LLMs to significantly advance the field of robotic motion planning and control.

3 Methods and Algorithms

3.1 Overview

This research explores the innovative integration of a Large Language Model (LLM), specifically the Google Gemini Pro Vision model hosted on Vertex AI cloud services[11], with teleoperation techniques for robotic manipulation. The objective is to use a slim black bar, mimicking the dVRK endowrist, to push a red cube towards a green cube within a simulated environment in Unity. The task capitalizes on the LLM's exceptional capabilities in information extraction and logical reasoning, relying on visual inputs from cameras and carefully crafted prompts to guide the robot's actions.

3.2 Directional Reasoning

The task is bifurcated into two sequential steps, with the initial phase dedicated to discerning the spatial configuration of the cubes and the robotic arm. The LLM is queried with images from the cameras to understand the relative positioning of the red and green cubes and the bar's orientation concerning the red cube. This is achieved through prompts designed to extract one-dimensional spatial relationships (left/right and above/below) from the LLM's analysis of the images. The simplicity of requiring a single-word response ("left," "right," "above," or "below") minimizes post-processing complexities and streamlines decision-making.

To ensure reliability in the LLM's output, given its probabilistic nature, each directional query is repeated five times. The majority vote from these iterations determines the final decision, thereby reducing the risk of erroneous directions due to sampling variability.

3.3 Direction Determination

Once the spatial configuration of the cubes and the robotic arm is ascertained, the subsequent step involves determining the precise direction for the arm's movement to effectively manipulate the red cube towards the green cube. This process entails aligning the robotic arm, represented as a bar, with the imaginary line that connects the two cubes, positioning it specifically at the red cube's end 2.

If the bar's orientation with respect to the red cube is congruent with the red cube's orientation relative to the green cube, the movement is straightforward— the bar proceeds in a direction that negates any extraneous spatial dimension, followed by moving towards the green cube. However, in instances where this alignment does not exist, a three-step maneuver is necessitated. Initially, the bar is repositioned to the outer extremity of the red cube, keeping the extraneous spatial dimension. Then the problem is reduced to the former scenario.

For illustrative purposes, consider a scenario where the red cube is positioned to the left of the green cube, and the bar is located below and to the left of the red cube. In this configuration, the bar needs only to ascend vertically to align with the cubes, thereafter moving horizontally to the right, guiding the red cube towards its target. Conversely, should the bar be situated below and to the right of the red cube, it necessitates an initial lateral movement to the left, surpassing the red cube, before assuming the alignment and push strategy.

Algorithm 1 Determine Direction for Bar Positioning

Require: firstMoveDir, firstMoveDir global.

{Input parameters}
$barPosX \leftarrow$ "whether the bar is on the left or right of the red cube."
$barPosZ \leftarrow$ "whether the bar is on the above or below of the red cube."
$cubePos \leftarrow$ "whether the red cube is on the left or right of the green cube."
$firstMoveDir \leftarrow$ "Direction to go in the first step"
$secondMoveDir \leftarrow$ "Direction to go in the second step"

{Returns whether we need the additional first step to move the bar to the outer side of the red cube.}
function DETERMINEDIRECTION($barPosX$, $barPosZ$, $cubeRelativePos$)
  if $cubeRelativePos$ == "left" **then**
    if $barPosX$ == "right" **then**
      $firstMoveDir \leftarrow$ INVERSEDIRECTION($barPosX$)
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosZ$)
      return $true$
    else
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosZ$)
      return $false$
    end if
  else if $cubeRelativePos$ == "right" **then**
    if $barPosX$ == "left" **then**
      $firstMoveDir \leftarrow$ INVERSEDIRECTION($barPosX$)
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosZ$)
      return $true$
    else
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosZ$)
      return $false$
    end if
  else if $cubeRelativePos$ == "above" **then**
    if $barPosZ$ == "below" **then**
      $firstMoveDir \leftarrow$ INVERSEDIRECTION($barPosZ$)
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosX$)
      return $true$
    else
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosX$)
      return $false$
    end if
  else
    if $barPosZ$ == "above" **then**
      $firstMoveDir \leftarrow$ INVERSEDIRECTION($barPosZ$)
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosX$)
      return $true$
    else
      $secondMoveDir \leftarrow$ INVERSEDIRECTION($barPosX$)
      return $false$
    end if
  end if

3.4 Movement Strategy

Upon establishing the correct orientation, the system then transitions to executing the movement. Direct LLM queries for precise movement metrics and subsequent one-step directly to the correct position were avoided due to the model's limitations in generating accurate numerical outputs. Instead, a teleoperation approach is adopted, where the robotic arm advances in small, discrete increments (0.01f units) toward the target position.

Termination criteria for movement are meticulously defined to avoid premature cessation due to the LLM's challenges in discerning spatial relationships at close proximities. Instead of relying on the same dimensional queries, the system shifts to orthogonal dimensions to ascertain when the robotic arm or the red cube has reached its intended position. For example, if movement is along the left-right axis, the termination query might assess the vertical alignment ("Is the bar directly below the red cube?") to ensure precise positioning. A similar method is adopted in the final push stage, where the query "Has the red cube touched/reached the green cube?" determines the completion of the task.

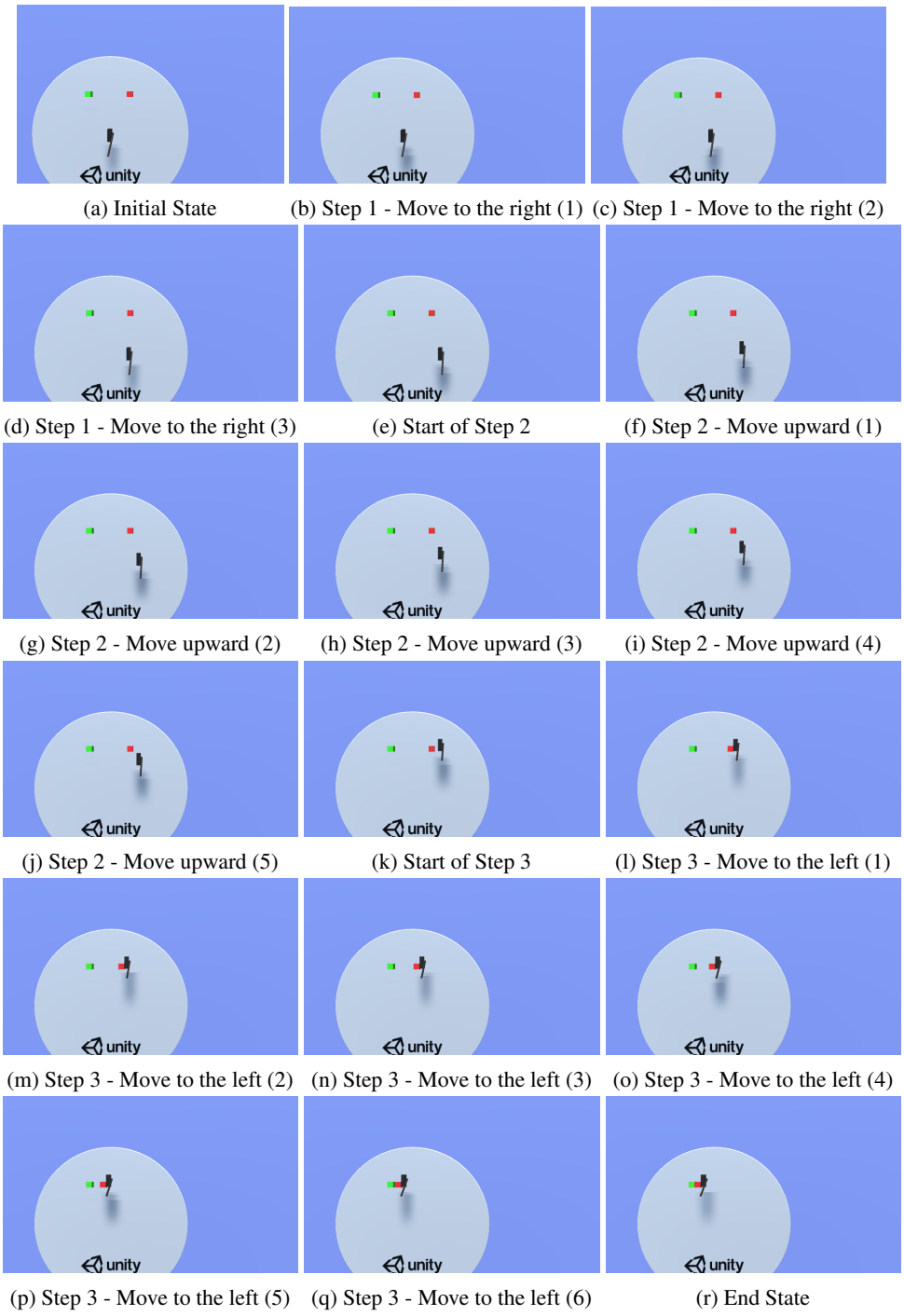
Algorithm 2 Step Logic for Teleoperation with LLM Feedback

Require: CamView is a Camera object.
Require: DirectionToMove is an object of type Dir.
Require: terminationQuery is a string containing the LLM query.
Require: adjustment is an integer representing manual adjustment magnitude.
{Function that controls the teleoperation based on LLM feedback}
function STEPLOGIC($CamView$, $DirectionToMove$, $terminationQuery$, $adjustment$)
  Data ← empty string
  moveSanity ← 0
  repeat
    Move stick in the direction specified by DirectionToMove
    Data ← Capture the current camera view
    Communicate with Gemini LLM using the terminationQuery and captured Data
    if (llmOutput equals "yes") AND (moveSanity is less than 2) **then**
      llmOutput ← "no"
      Log error: "Sanity check captured error output"
    else if (llmOutput equals "no") AND (moveSanity is greater than 36) **then**
      Log error: "Sanity check captured too much iteration"
      break
    else
      Log response from LLM
    end if
    Increment moveSanity
  until llmOutput does not equal "no"
  if adjustment is not equal to 0 **then**
    Move bar in the current direction by adjustment amount
  end if

4 Experiments And Ablation Tests

4.1 Exexcution

The program queries the following two camera renderers with three prompts:
"Given the image with a black bar and a red cube, is the black bar on the left or right of the red cube? Please respond with only one word: 'left', or 'right'"
Given the image with a black bar and a red cube, is the black bar on the above or below the red cube? Please respond with only one word: 'above', or 'below'"
"Given the image with a red object and a green cube, is the red object on the left or right of the green cube? Please respond with only one word: 'left', or 'right'"
After successfully outputting the spatial relationships, the program will shift the stick next to the red cube and push it toward the green cube. The time series of execution is provided below.



(a) Initial State (b) Step 1 - Move to the right (1) (c) Step 1 - Move to the right (2)

(d) Step 1 - Move to the right (3) (e) Start of Step 2 (f) Step 2 - Move upward (1)

(g) Step 2 - Move upward (2) (h) Step 2 - Move upward (3) (i) Step 2 - Move upward (4)

(j) Step 2 - Move upward (5) (k) Start of Step 3 (l) Step 3 - Move to the left (1)

(m) Step 3 - Move to the left (2) (n) Step 3 - Move to the left (3) (o) Step 3 - Move to the left (4)

(p) Step 3 - Move to the left (5) (q) Step 3 - Move to the left (6) (r) End State

Figure 2: Time Series Documentation of the Program Execution

4.2 Ablation Test

4.2.1 Cube-Cube Camera

During the experiments, it is obvious that given a low FOV (Field of View), the image is too narrow to capture both cubes. If the resolution is too high, it may capture too many unnecessary pixels to overwhelm the LLM. this ablation test will experiment with different scaling(Field of View) and

resolution combinations to give the best output. The Bar has been relocated next to the red cube for convenience, and the rest is the same as in Figure 1.

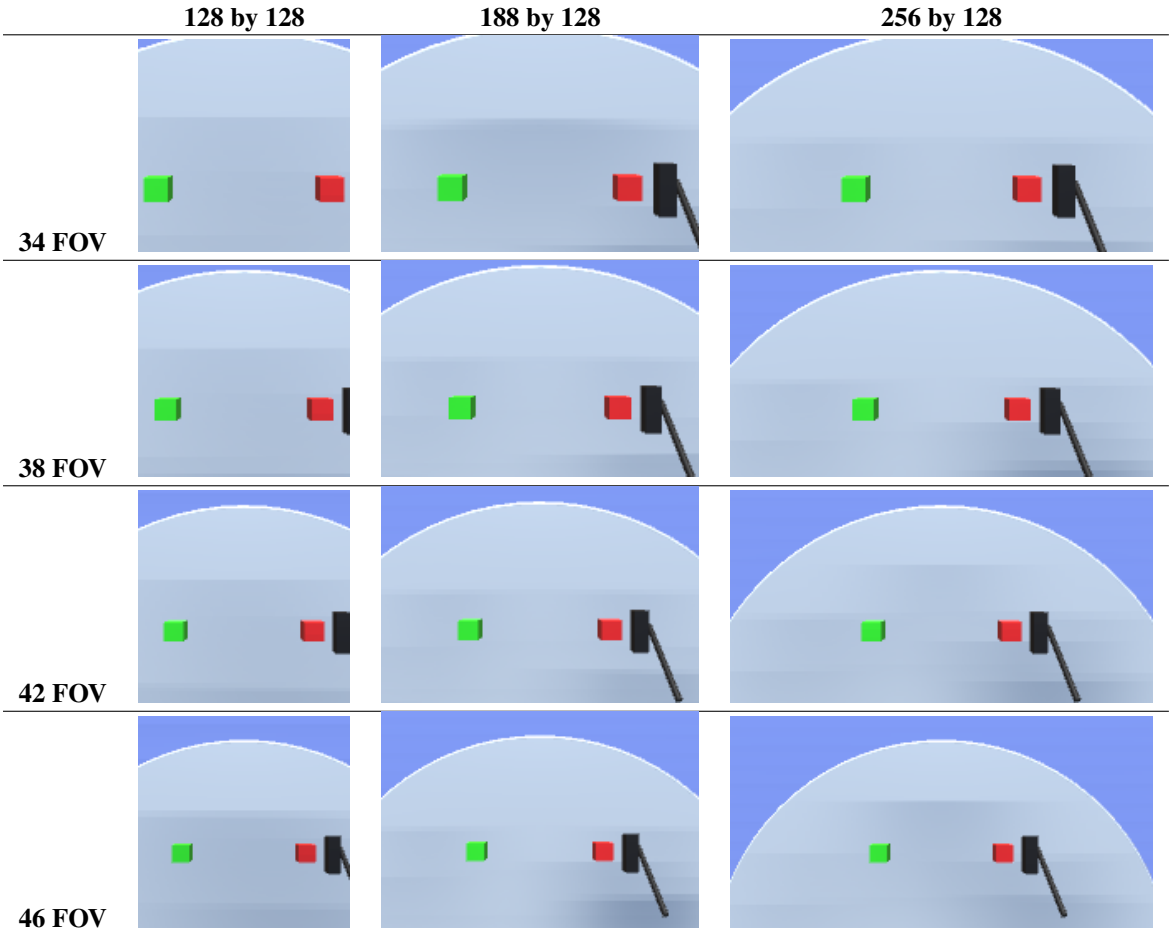


Figure 3: Camera FOV and Resolution Comparison

First is to test with which of the camera setup, LLM can output the correct output for "whether the red cube is on the left or right of the green cube." The query for each image iterates 15 times and the number of correct answers "right" outputted is recorded.

	128 by 128	**188 by 128**	**256 by 128**
34 FOV	15	15	15
38 FOV	15	12	15
42 FOV	0	15	15
46 FOV	15	15	15

Table 1: Number of correct answers replied

Based on the result, combinations of 38 FOV 188 by 128, and 42 FOV 128 by 128 are omitted, and the remaining are tested for shifting the bar to push the red cube to the green cube without sanity check intervention or premature termination. We perform 5 tests on each case and record the number of successes.

Explanation: In each slot, the first input records the number of successes. If not 5, then the subsequent records the kind of Failures (U: reaching the upper bound/Unable to detect termination, L: Did not pass lower bound/Unable to start, P: Premature termination/ stop half of the way). Here the upper bound is set to 37 because that is when cubes will land at the edge of the table, and any

	128 by 128	188 by 128	256 by 128
34 FOV	4:U	5: 21	4:U
38 FOV	2:U	omitted	1:U
42 FOV	omitted	1:U	1:U
46 FOV	3:U	1:U	0:U

Table 2: Number of correct Executions

further push will drop objects off. If the first number is 5, then the subsequent records the number of iterations required to stop. (it should be at least 19, where the first contact between cubes appears, and the smaller the better)
Thus, 34 FOV with 188 by 128 resolution is the best choice. Further observations show that if the width is set to 128, there is not much space to the left of the green cube, and thus little room for error. If the LLM cannot detect termination within one or two rounds, the objects are out of view. If the width and FOV are set too high, then unnecessary information is captured, hence LLM will need to process more bytes, deteriorating its capability. Also with a small FOV, the camera and LLM can focus better on the subject of matter and produce better outputs.

4.2.2 Translation of Camera

Interested in how bad the camera views with higher FOV work, an experiment is conducted on how changing other parameters will affect the LLM output of both queries. The 42 and 46 FOV camera series is chosen and the x position of the camera is studied. Knowing the view of 128 by 128 resolution is too limited, only the last two resolutions are included.

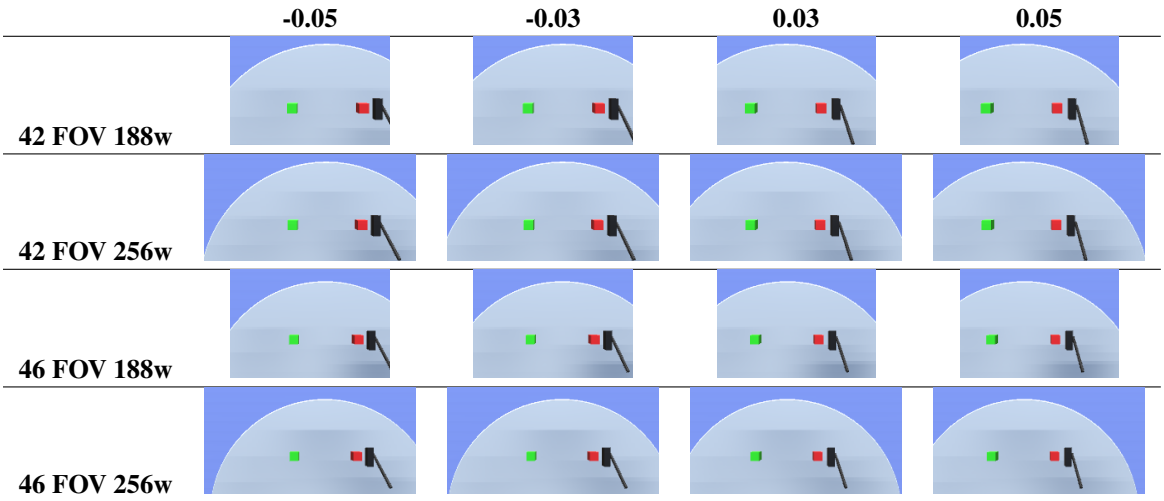


Figure 4: Camera FOV and Translation Comparison

Step one is to test with which of the camera setup, LLM can output the correct output for "whether the red cube is on the left or right of the green cube." The query for each image iterates 15 times and the number of correct answers "right" outputted is recorded.

	-0.05	-0.03	0.03	0.05
42 FOV 188w	15	15	15	15
42 FOV 256w	15	15	15	15
46 FOV 188w	15	15	15	15
46 FOV 256w	15	15	15	15

Table 3: Number of correct answers replied

This shows the shift in the x-axis does not affect the LLM's judgment on the positioning of objects. What about whether the LLM can terminate the process at the desired locations? Similar tests were

performed, and the results are collected in the following table.

	-0.05	**-0.03**	**0.03**	**0.05**
42 FOV 188w	4:U	4:U	2:U	1:U
42 FOV 256w	5:23	4:U	0:U	1:U
46 FOV 188w	1:U	2:U	1:UP	1:U
46 FOV 256w	3:U	2:U	1:U	0:U

Table 4: Number of correct answers replied

This experiment shows that shifting the camera slightly to the left significantly improves the performance while remaining capable of reasoning about spatial relationships.
The reason is, that by shifting left, the target object (the green cube) will be located more toward the center of the camera which facilitates reasoning. Furthermore, it leaves more space on the left, which gives LLM additional changes to output "wrong" answers and retry, effectively improving fault tolerance. On the opposite, by shifting right, the LLM is inaccessible to these improvements. The target object (the green cube) is located more toward the periphery, and the LLM must make correct decisions in fewer steps. Further observations show that the termination step required by 42 and 46 FOV is more than what is required by 34 FOV in the previous section, which is reasonable because the camera with 34 FOV is much more focused, enlarging objects and revealing more details.

4.2.3 Apparatus Positions

The third test studies both the two bar-cube cameras and the direction determination logic. Cameras with different widths (128 and 188) are used to test the bar located at both left below and right below the red cube. The goal in both the left and right scenarios is to shift the bar towards the red cube on the far side. In addition, in the case of "right below", no horizontal translations (ie. step 1) should be needed.

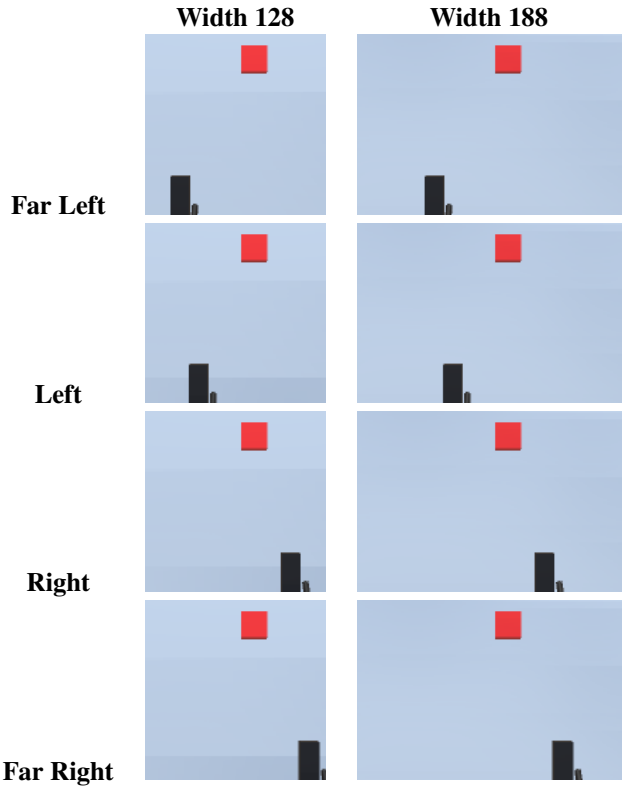


Figure 5: Arrangement of photos by bar position and camera width.

The following tests the LLM's ability to determine "left", or "right" correctly.

	Width 128	Width 188
Far Left	15	15
Left	15	15
Right	9	15
Far Right	15	15

Table 5: Number of correct answers replied under two conditions.

It is obvious, that if the bar is only slightly right of the red cube, then with a small width (small width is unable to stretch the view to amplify the rightness), the LLM will have a hard time distinguishing the positional relationships.
lastly, for each of the 8 scenarios, the test is performed to move the bar form either left below or right below to the direct right of the red cube. In particular, the horizontal and vertical movements are tracked separately, where the Right and Far Right cases should not invoke any horizontal movements.

	Width 128	Width 188
Far Left	5	4:U
Left	5	4:U
Right	0	0
Far Right	0	0

Table 6: Horizontal movement

Notice both the Right and Far Right scenarios never invoke this step as intended.

	Width 128	Width 188
Far Left	5	4:P
Left	5	4:P
Right	5	5
Far Right	5	3:P

Table 7: Vertical movement

Having a width of 128 will outperform a width of 188 in both movements. This is because width 128 provides LLM with a narrower view, terminating the bar early in the horizontal movement. This not only reduces the possibility of overreach error in horizontal translation but also provides a better starting point for the subsequent vertical movement. Starting the vertical movement with the bar closer to the red cube in the x-axis (ie when only considering the "left-right" dimension, the bar is closer to the red cube) results in better performance.

4.2.4 Randomize Path

The final experiment is about how different shapes might affect the path objects take. Two experiments are performed on changing only the red cube to a red disk and changing both the red cube and the base of the robot arm to disks. The latter one has an interesting behavior which will be presented in this paper.

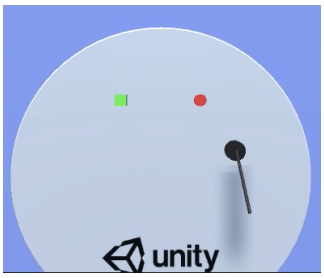


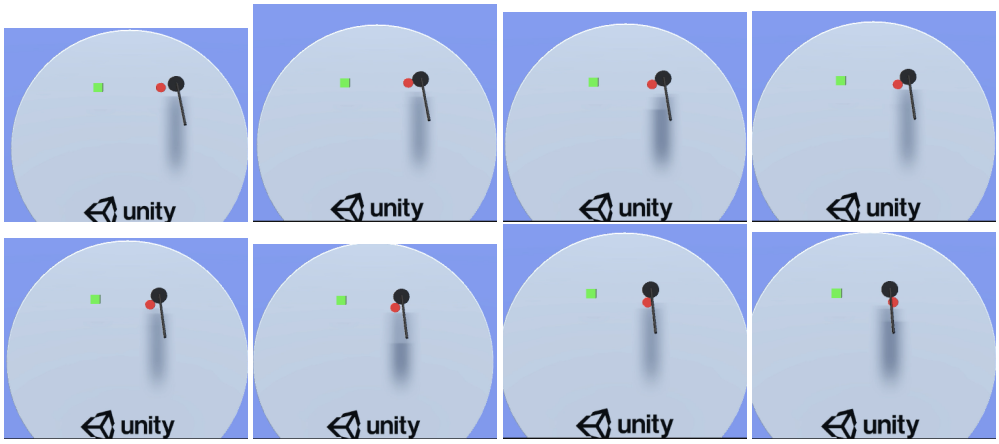
Figure 6: Double Disk Setup

Since the two objects are both disks, they will follow a straight path only if the middle of both disks is touching. The experiment shows that with even one step difference (0.01 unit), the path would be angled, and as further steps amplify the error, the two objects will very quickly lose contact. (See Figure 7 7)

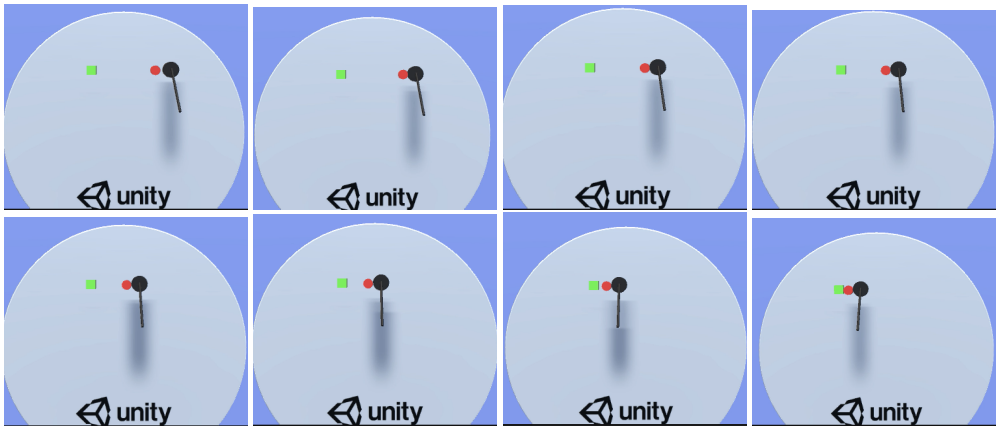
Algorithm 3 Smart Teleoperation

function SMARTSTEPLOGIC($CamA$, $CameraM$, $DirToMove$, $termQuery$, $adjustment$)
  Data, DataAlign ← empty string
  moveSanity ← 0
  repeat
    Move stick in the direction specified by DirToMove
    DataAlign ← Capture form $CamA$
    Communicate with Gemini LLM using the AlignmentQuery and DataAlign
    if llmOutput equals "above" **then**
      Move stick in the direction opposite of DirToMove
      Move stick above
    else if llmOutput equals "below" **then**
      Move stick in the direction opposite of DirToMove
      Move stick below
      Data ← Capture form $CamM$
      Communicate with Gemini LLM using the termQuery and captured Data
      if (llmOutput equals "yes") AND (moveSanity is less than 2) **then**
        llmOutput ← "no"
        Log error: "Sanity check captured error output"
      else if (llmOutput equals "no") AND (moveSanity is greater than 36) **then**
        Log error: "Sanity check captured too much iteration"
        break
      else
        Log response from LLM
      end if
      Increment moveSanity
     llmOutput does not equal "no"
    if adjustment is not equal to 0 **then**
      Move bar in the current direction by adjustment amount
    end if

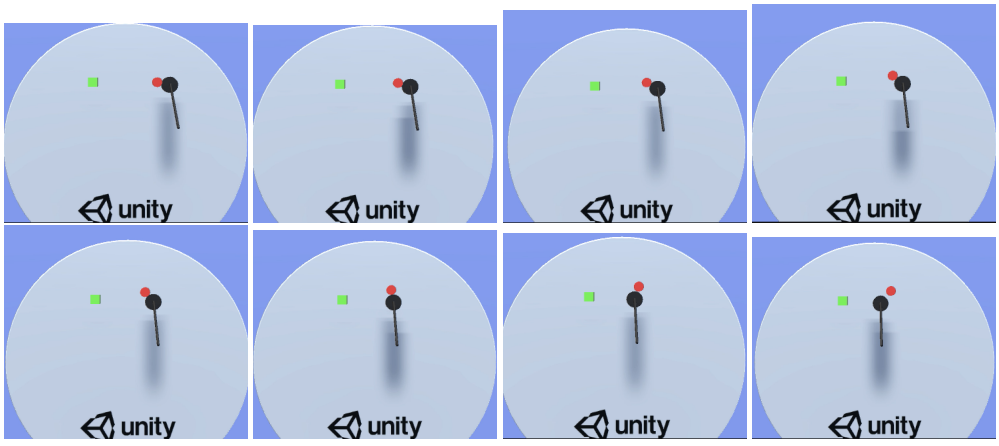
To resolve this problem, a Smart Teleoperation algorithm is proposed. In each iteration, it queries Gemini if the red disk stays at the same height as the apparatus. The LLM will reply with "same", "above", or "below". This detection is a bit delayed as the ill-alignment must be big enough to be discovered, so the apparatus needs to be moved back a little, then



(a) Fail by shifting downwards



(b) Success



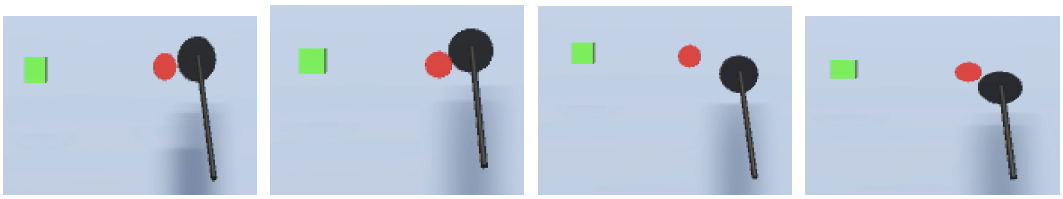
(c) Fail by shifting upwards

Figure 7: Time Series Documentation of Double Disks

move up in case of "above" or move down in case of "below" so that it does not bump into the red disk.

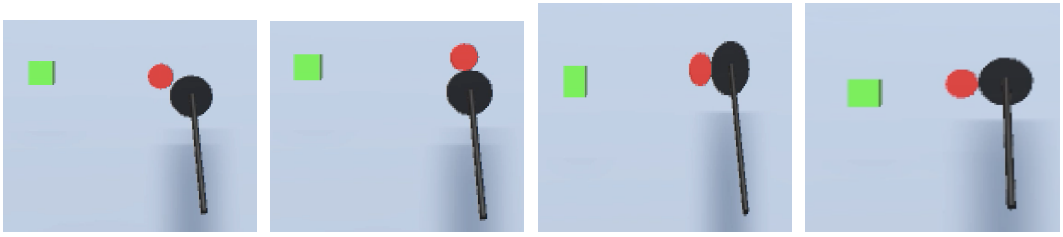
This modification solved this problem very well. In particular, if the error accumulated from the first two steps results in the incorrect starting position of step 3 (ie, the apparatus started way below or way above the red disk), then this algorithm will detect this almost immediately, and within several iterations, the apparatus will be moved onto the same height as the red disk, comparing to the previous

algorithm which is completely impotent to this kind of situation.
Thus, with this new proposal, the program not only can handle the misalignment raised in the middle of step 3 but also gives considerable room for error in the first two steps, as long as they land the apparatus around the red disk, rest will be gracefully handled by smart teleoperation.



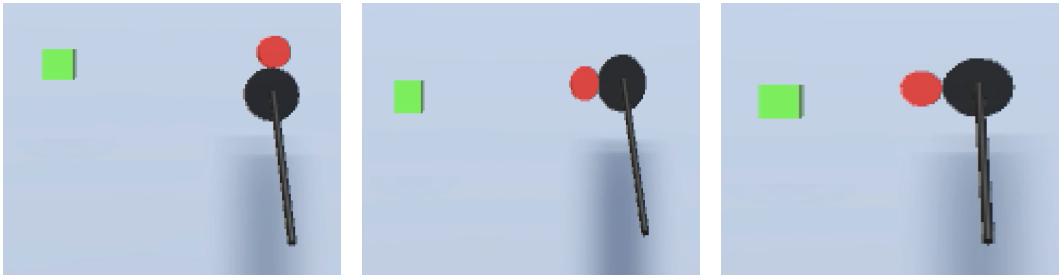
(a) Misalignment (b) Red disk slightly below (c) Relocation (d) Resume execution

Figure 8: Red disk shift down during execution



(a) Misalignment (b) Red disk slightly above (c) Relocation (d) Resume execution

Figure 9: Red disk shift up during execution



(a) Incorrect start (b) Immediate relocation (c) Correct execution

Figure 10: Incorrect start positioning

5 Discussion

In the dynamic field of robotics, the integration of Large Language Models (LLMs) is a growing trend, heralding new capabilities as seen in recent academic explorations. Abhilash Pandya's work[6] is one such example, offering an upgrade to the da Vinci Surgical Robot by equipping it with a ChatGPT-based natural language interface that supports multiple languages, enhancing the surgeon's interaction with the robot for improved precision and safety. While their work provides a foundational AI integration with the da Vinci system, it stops short of including empirical user studies.

Similarly, Sai Vemprala et al. have presented a methodological framework for embedding ChatGPT within robotic systems[10]. Their approach highlights a multifaceted pipeline that utilizes diverse prompting strategies to understand human instructions and makes function calls on an abstract function library that can adapt to different robot APIs, ultimately completing impressive tasks like performing basic housework and examining power plants.

This paper advances the discourse by presenting a practical application of LLM-Robotics integration that proves its efficacy in real-world tasks and provides a modified HTTP client connection that facilitates communication with cloud-based LLMs like Gemini, broadening the scope beyond single-model reliance. This project innovates with a teleoperation mechanism that implements step-by-step movement through LLM guidance, offering granular control that enhances fault tolerance and reduces the need for precise coordinates.

Crucially, this research diverges from the prevailing use of LLMs solely as a user interface or a high-level decision-maker. Instead, the LLM is embedded within the robot's core operational logic, thus streamlining the action process and providing the robot with some degree of common sense and logical reasoning capabilities. This integration eliminates the complexity of traditional rule-based driving code and positions LLMs as an intrinsic element of the low-level control system, enabling the robot to make nuanced adjustments autonomously.

6 Conclusion

This study presents an innovative approach to enhance robotic manipulation by integrating Large Language Models (LLMs) and teleoperation techniques, addressing the challenges posed by uncertain environments. The method demonstrated through a simulation involving an abstract representation of a da Vinci Research Kit (dVRK) endowrist maneuvering objects within a Unity-based environment, highlighted the potential of LLMs, particularly the Google Gemini model, to iteratively guide robotic actions with enhanced fault tolerance and adaptability. The experiments, which included variations in camera perspectives, object placements, and shapes, further expose the characteristics of the Google Gemini Model and how it interacts with robots. By abstracting the robotic endowrist to a simplified form, relying on visual inputs for spatial reasoning, and executing the LLM-equipped teleoperation algorithm, this project demonstrates successful navigation within complex dynamic environments without the need for precise destination coordinates.

Future research may explore the potential of varying the underlying Large Language Models (LLMs), such as employing models like ChatGPT or LLaMa, to assess their efficacy in guiding robotic manipulation in uncertain environments. Additionally, the integration of virtual reality (VR) techniques holds promise for enhancing the abstraction process, enabling a more intuitive mapping back to a physical da Vinci Research Kit (dVRK) endowrist. Exportations of successful simulations to real-world scenarios may also present a significant step forward, promising to bring theoretical advancements into tangible, operational robotics systems, thereby expanding the horizon of robotic manipulation and its applicability in dynamic, unpredictable settings.

References

[1] Rohan Anil et al. "Gemini: A Family of Highly Capable Multimodal Models". In: *arXiv preprint arXiv:2312.11805* (2023). DOI: 10.48550/arXiv.2312.11805. URL: https://doi.org/10.48550/arXiv.2312.11805.

[2] Google DeepMind. *Language to Rewards: Enabling Precise Task Specification in Reinforcement Learning*. https://language-to-reward.github.io/. Accessed: 2024-03-19. 2023.

[3] JHU dVRK. *Large Needle Driver 420006 - High Resolution CAD*. https://github.com/jhu-dvrk/instrument-cad/tree/main/Large_Needle_Driver_420006/high_res. 2023.

[4] Kevin Lin et al. "Text2Motion: Generating Diverse and Natural Text-Driven Motion with LLMs". In: *arXiv preprint arXiv:2303.12153* (2023). URL: https://arxiv.org/abs/2303.12153.

[5] OpenAI. *ChatGPT-4: Enhancements and Innovations*. https://openai.com/blog/chatgpt-4. Accessed: 2024-03-19. 2023.

[6] Abhilash Pandya. "ChatGPT-Enabled daVinci Surgical Robot Prototype: Advancements and Limitations". In: *Robotics* 12.4 (2023), p. 97. DOI: 10.3390/robotics12040097. URL: https://www.mdpi.com/2218-6581/12/4/97.

[7] ROS Planning. *MoveIt: The MoveIt Motion Planning Framework*. https://github.com/ros-planning/moveit. Accessed: 2024-02-01. 2024.

[8] Chan Hee Song et al. "LLM-Planner: Leveraging LLMs for Task Planning in Robotics". In: *arXiv preprint arXiv:2212.04088* (2022). URL: https://arxiv.org/abs/2212.04088.

[9] Unity Technologies. *Unity Robotics Hub: Pick and Place Tutorial*. https://github.com/Unity-Technologies/Unity-Robotics-Hub/tree/main/tutorials/pick_and_place. 2023.

[10] Sai Vemprala et al. *ChatGPT for Robotics: Design Principles and Model Abilities*. Tech. rep. MSR-TR-2023-8. Microsoft, Feb. 2023. URL: https://www.microsoft.com/en-us/research/publication/chatgpt-for-robotics-design-principles-and-model-abilities/.

[11] Jack Code Wu. *GeminiAI.Net*. https://github.com/jackcodewu/GeminiAI.Net. Accessed: 2024-02-22. 2023.