STEP: State-Aware Task Estimation and Planning with Multi-Modal LLMs for Human-Robot Collaboration

Maitrey Gramopadhye[1], Prakash Baskaran[2], Xiao Liu[2], Songpo Li[2] and Soshi Iba[2]

***Abstract*— Effective human-robot collaboration in industrial settings requires robots to understand human intentions and assist with task planning, reducing workload. Recent works have explored the use of Multi-modal Large Language Models (MM-LLMs) for task planning in such data-scarce scenarios, leveraging in-context learning to interpret user actions and generate long-horizon action plans in natural language. However, MM-LLMs inherently lack an understanding of system states and do not track state transitions, often leading to hallucinated actions that deviate from the intended goal. Additionally, generating action plans in natural language tends to limit the generated plans to a high level, introducing ambiguity in action execution. To address these limitations, we propose the State-aware Task Estimator and Planner (STEP), which prompts a MM-LLM to explicitly estimate the state of the system and predict the state transitions resulting from executed actions. By forecasting future states alongside actions, STEP ensures task-convergent planning while also providing additional assistance parameters necessary for executing the predicted actions. We evaluate STEP in a simulated environment using a robot assembly task. Our approach outperforms the state-of-the-art by 32.8% in action executability and 14.8% in final-state error.**

I. Introduction

As robots have become more affordable and versatile, their applications have expanded, particularly in industries requiring greater human interaction and generalization. Human-robot collaboration improves efficiency by drawing on their respective strengths. However, effective teamwork requires shared control and mutual understanding. For seamless collaboration, robots must be able to infer human intentions and take over some of the workload. Relieving control over to a robot can enable a person to supervise multiple robots or take on a creative role by developing a vision and delegating execution to the robot [1]. However, several tasks in which robot assistance would help are intrinsically human and designed for human-centric environments, and training robots to infer human goals and generate actionable plans requires extensive data collection, which remains a significant challenge [2]–[4]. An alternative approach that has recently gained popularity is the use of MM-LLMs for long-term action anticipation (LTA) from observations [5]–[8]. Prior work has explored the in-context performance of MM-LLMs to infer a human's goal and predict the future actions needed to achieve it, based on the sequence of actions performed by the human [9]. However, directly querying an MM-LLM for planning robot actions has several limitations.

This work was done during an internship at Honda Research Institute
[1]University of North Carolina at Chapel Hill, United States
[2]Honda Research Institute, San Jose, CA, USA
Correspondence email: maitrey@cs.unc.edu

Since MM-LLMs lack real-world interaction during pre-training, they struggle to track the effects of their planned actions [10]–[12]. As a result, an agent that executes the generated actions often deviates from the intended goal state. To address this, recent work has proposed explicitly querying MM-LLMs to track the state of the system, allowing them to measure progress toward the goal [13]–[17]. However, these approaches either rely on assumptions such as the ability to comprehensively describe system states in natural language or affordably query the environment for state information; or they constrain their focus by treating state tracking separately from task planning. In this paper, we extend prior research on LTA by relaxing some of the assumptions and constraints to better match a real-world industrial setting.

Another key challenge in applying MM-LLMs to robotics is handling the high dimensionality of the action space. Since MM-LLMs fail to consider the nuances of real-world execution, several previous works limit their output to a cursory action plan of "verb, noun" pairs (e.g., "pick, cup", "pour from, cup", "place on, table"), while relying on predefined functions to abstract away the other parameters [11], [14], [18]–[20]. However, this often leads to ambiguity during action execution. For instance, executing "place on, table" with a cup requires determining a placement location on the table, while accounting for other objects, and the orientation of the cup after placement (upright, upside down, etc.). Some works mitigate this by incorporating feedback, either from humans in the loop or by querying the environment, and re-planning for any failures.

In this paper, we propose STEP to explicitly estimate and propagate a structured representation of the environment state. Specifically, we consider an ubiquitous industrial scenario in which an operator is assembling a structure on a workstation and wishes to hand over the planning for building the structure to a robot. Our mock task serves as a proxy for real-world industrial tasks, such as machine assembly or modular fixture construction, where human-robot collaboration would be beneficial. As shown in Fig. 1, we use the actions already performed by the operator and an image of the workstation, along with a small curated set of in-context examples, to prompt an MM-LLM to estimate - the overall structure the operator is trying to build (*task estimation*) (See §III-A) and a structured state representation of the workstation in a JSON format (*state representation generation*) (See §III-B), followed by the remaining actions required to construct the structure (*action prediction*) (See §III-C) and the future states of the environment after executing the proposed actions (*state prop-*

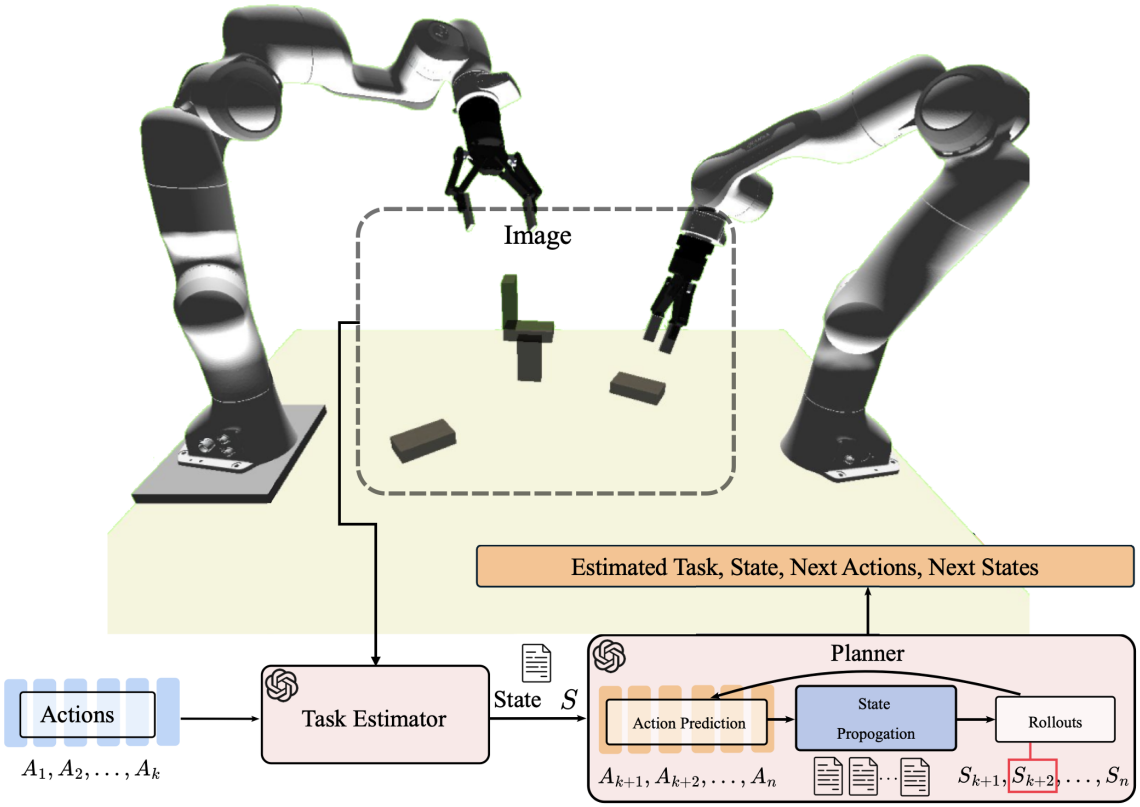

Fig. 1: **STEP with robot system setup.** Predicting structured state representations along with future actions allows STEP to quantitatively measure the distance of a state from the goal and deduce assistance parameters required for executing the predicted actions.

agation) (See §III-D), without requiring any interaction with the environment. Propagating a structured representation of the environment allows us to predict assistance parameters required to complete the predicted actions that could not be expressed as a "verb, noun" pair. It also allows us to track the distance from the inferred goal quantitatively. We use this distance metric to truncate and regenerate the future action plan and states iteratively, to ensure progress towards the inferred goal (*rollouts*) (See §III-E). STEP employs an MM-LLM for explicitly estimating and predicting a structured state representation to support robotic action planning in an industrial setting. Such industrial settings (1) offer a valuable opportunity for human–robot collaboration, (2) are inherently structured, making our choice of a structured state representation well-suited, and (3) remain underrepresented in existing MM-LLM pre-training datasets, posing unique challenges for model generalization.

We evaluated our approach on a dataset from a simulated block assembly scenario, in which human operators teleoperate two Franka robot arms situated on either side of a table (See §IV-A) [21]. The goal of the assembly scenario is to construct a structure using five identical wooden blocks. We use several metrics (See §IV-B), including executability, longest common subsequence (LCS), and final-state error to test our generated action plans. Overall, our method increased action executability by 32.8% and decreased final-state error by 14.8%, compared to a state-of-the-art baseline (See §V).

II. Related Work

STEP is inspired by papers on shared autonomy and LTA using LLMs. We address some of their limitations by extending prior work on tracking a state representation of the environment for LTA.

A. Shared autonomy in teleoperation

The idea of shared teleoperation has gained popularity in industry as a way to increase production while empowering humans to assume a supervisory role [1]. Previous work has approached shared teleoperation by having a semi-autonomous robot provide assistance to an operator [22]–[24]. Usually, the robot starts as non-autonomous and attempts to infer the user's goal [25]–[28], or form a reasonable estimate of it [29], by observing their actions. The robot then provides assistance to complete the task by taking control [30] or providing visual, haptic, or multi-modal cues [24], [31]–[33]. In this paper, we follow a similar pipeline, by taking as input the actions performed by the user as a sequence of natural language actions (e.g. "Pick block 1", "Stand block 1", "Withdraw") and predicting the user's goal, followed by the actions required to complete the goal.

B. Large language models and long-term action anticipation

Large Language Models (LLMs), and their images + text counterparts, MM-LLMs, have shown great performance

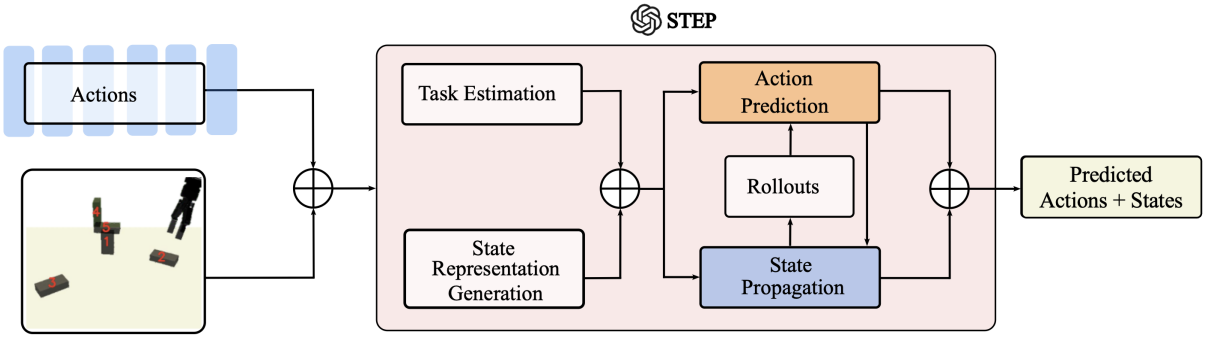

Fig. 2: After a hand-over by the operator, the overall task and current system state are estimated. They are then used to iteratively rollout next actions and states multiple times. Each rollout selects the closest state to the goal to guide actions towards the estimated task.

on diverse tasks [34]–[40]. Particularly useful is their in-context learning ability, where LLMs used as frozen models can generalize across domains when few domain-specific examples are provided during inference [41], [42]. Several studies have proposed using in-context learning to ground LLM output in robot-actionable steps [11], [18]–[20]. In-context learning is especially useful in robot task planning for industry, as LLMs are often not trained on data that is relevant for these scenarios. Like shared autonomy planning, LTA involves forecasting an agent's future actions from an initial sequence of observed behavior [5]–[8]. Previous research has explored converting a video into a sequence of natural language actions and using that as input to LLMs to predict future actions. AntGPT [9] proposes a method in which an LLM predicts the overall goal of an agent from a sequence of natural language actions and then generates the remaining actions required to achieve that goal. VidAssist [43] proposes to take as input the initial sequence of actions and the overall goal to iteratively build a tree of possible future action sequences and find the sequence that will best satisfy the given goal. Inspired by these approaches, our prediction pipeline further extends this framework by also predicting and propagating the state of the environment.

C. State estimation using language models

LLMs often struggle with planning and reasoning tasks. Being trained for auto-regressive sequence-to-sequence translation, LLMs lack any motivation to keep track of the state of a system [12]. As a result, LLMs often hallucinate actions that diverge from the intended goal. To overcome this limitation, previous research has explored using chain-of-thought reasoning to explicitly track the state of the system with LLMs [44], [45]. RAFA [13] proposes iteratively building a tree of action and state sequences, followed by utilizing these sequences to query an LLM to select the best action to perform. However, RAFA requires access to the initial ground-truth state of the system to begin planning. Similar to RAFA, LLM-State [14] and Kaige et al., [15] also output the future state representations in natural language to guide an agent's actions. However, a natural language state representation is often not expressive enough to represent the state of a robot workspace and only allows prior works to get a vague guidance towards the goal state. In our method, we query an MM-LLM to estimate the environment state and propagate it as a structured JSON object, allowing us to calculate a quantitative distance from the goal state. Statler [16] proposes querying an LLM to infer and update the state of a system as a structured representation. However, Statler does not consider the problem of action planning and only propagates the state for one action. Wall-E [17] proposes to predict future actions and propagate a state representation. However, it needs the initial ground-truth state of the system as input and an exploratory phase to learn the rules about the world, which might not be feasible in the real world.

III. Approach

STEP takes as input the actions performed by an operator and an image of the workstation, along with a set of example interactions to prompt an MM-LLM. As illustrated in Fig. 2, our pipeline predicts the overall task the operator is trying to accomplish. Simultaneously, it also estimates a structured state representation of the current workspace from its image. It then uses the estimated task and action history to predict future actions required to complete the task. The current state is then propagated to estimate the future states of the workstation. Using the structured future state representations, it calculates their distance from the goal state of the estimated task and truncates the future actions at the state with the least distance. STEP also repeats the action prediction and state propagation steps for multiple rollouts to iteratively get closer to completing the task.

A. Task Estimation

Following prior work [9], we employ a top-down approach by first estimating the overall task (T) that the operator wants to complete. STEP takes as input the actions performed by the operator so far ($A_1, A_2, ..., A_k$) and the image of the current workstation (I) to prompt an MM-LLM for the task. In the prompt, we also include examples (of the format `(actions, <image>) -> Task`) from a small pre-collected dataset, and a list of all possible tasks (see §IV-A) to guide and constrain the MM-LLM output.

B. State Representation Generation

We prompt an MM-LLM with the image of the workstation (I) for a structured representation of the state (S). We follow a chain-of-thought approach to query the model multiple times, each time asking it to describe various parts of the workspace layout. First, we query it to get the locations (top right, top left, center right, etc.) of each of the 5 blocks in the image. These locations of the blocks in the image help us ground our subsequent prompts in the image, where we query the model to provide the following information for each block: (i) the block's placement location (including the number of the block it is placed on, if any) (ii) the block's orientation (i.e., standing, lying face down, or lying on side) (iii) the block's top face orientation (facing parallel or perpendicular to the camera) (iv) If the block is on another block, describe its location relative to the base block. We also include a brief description of the workstation and example images of the workstation in the prompts. In the end, we combine the collected information and ask an MM-LLM to format it as a JSON object (See Fig. 3 for more details).

For `block_id` in `{1,2,3,4,5}`:
Block `block_id`:

Block location:	{Table, Block (with number), Gripper}
Block orientation:	{Stand, Lie on face, Lie on side, NA}
Top face orientation:	{Parallel, Perpendicular, NA}
Specific location:	{Left, Center, Right, NA}

Fig. 3: For each of the blocks our structured state representation records the block location (table/another block with its number), block orientation (stand/lie on face/lie on side/NA), top face orientation (long-side parallel/perpendicular to camera/NA), specific location (left/right/center if the block is on another block, otherwise NA). The modular state design also allows extension to other scenarios by adding descriptors - 'nearby objects', 'facing', 'containing objects' etc.

C. Action Prediction

We use the estimated task (T) and the action history ($A_1, A_2, ..., A_k$) to prompt the MM-LLM for the future actions ($A_{k+1}, A_{k+2}, ..., A_n$) that need to be performed to complete the task. We also include examples (of the format - `performed actions -> future actions`) in the prompt, taken from the pre-collected dataset for completing the estimated task. During our experiments, we discovered that including multi-modal data in the prompt for this stage degraded the performance, so we did not use the state or image information for action prediction.

D. State Propagation

We propagate the estimated current state of the workstation (S) to track the changes resulting from executing the predicted actions. We prompt the MM-LLM using $A_1, A_2, ..., A_k$, S and A_{k+1}, to get as output the next state (S_{k+1}), also as a JSON object. We then iteratively set the propagated state as the current state and predict the states resulting from executing all predicted actions ($S_{k+1}, S_{k+2}, ..., S_n$). For example, getting S_{k+2} would involve - $A_1, A_2, ..., A_k, A_{k+1}$, S_{k+1}, A_{k+2} -> S_{k+2}. Examples of the same format relevant to the estimated task are also included in the prompt. During our experiments, we found that using a chain-of-thought approach to break down the state propagation into two parts improved performance. For propagating the state at each action, we first ask the MM-LLM to describe the changes in the state in natural language, and then use the description to get a JSON representation of the next state.

E. Rollouts

Using a structured state representation of the workstation allows us to manually calculate its distance (d_T) from the estimated task completion state. We calculate the distance as the total number of differences between the state and the task completion state. Since our state representation has 5 blocks and each block has 4 different characteristics, d_T can range from 0 to 20 (See Fig. 3). For every rollout, we query the MM-LLM for the future actions and states. We then calculate d_T for all states (current and predicted), and truncate the predicted actions and states after the state with the first instance of the lowest d_T (e.g. S_l could be the state with the lowest d_T). For the next rollout, we set S_l as the current state and use $A_1, ..., A_k, ..., A_l$ and S_l for action prediction and state propagation. Since S_l has a d_T equal to or lower than S, with each rollout STEP identifies a state closer to the inferred goal than the state at the start of the rollout, and helps guide the action plan towards the goal. We repeat the rollouts for $max\_rollout$ times, which is a hyperparameter, or until we reach a state with $d_T = 0$. After the final rollout, we return the predicted action sequence ($A_{k+1}, ..., A_l$) and state sequence ($S, S_{k+1}, ..., S_l$), where d_T of S_l is the lowest among all states.

IV. Evaluation

We compared our method with a state-of-the-art LTA baseline inspired by the top-down approach proposed by AntGPT [9]. The baseline consisted of using the same inputs as our method, for *task estimation* (§III-A) followed by a single rollout of *action prediction* (§III-C), with no state information. We selected this comparison to directly examine the impact of explicitly estimating and tracking the system state on action prediction accuracy. To isolate and study this effect, we deliberately excluded potential confounding factors such as interactive planning or human-in-the-loop feedback. Also, several related works we examined impose constraints such as requiring access to the initial ground-truth state of the system for planning, an assumption that is often unrealistic in real-world industrial human–robot collaboration scenarios. Moreover, many approaches address only specific subcomponents of the overall action planning problem rather than the complete pipeline. Considering these factors, we identified AntGPT [9] as a suitable inspiration for our baseline.

A. Experimental Setup and Dataset

We evaluated our method on a dataset of teleoperated manipulation sequences collected by Baskaran et al. [21]. The collected manipulation sequences involved users teleoperating two robots in real time within a simulated environment using HTC Vive Pro controllers [46]. The simulated workstation composed of a table mounted with two Franka Emika robot arms [47] on either side, and five identical wooden block assembly pieces. The collected dataset had 495 instances from 19 users, where each instance involved operators accomplishing the goal of assembling the blocks into one of the 8 different structures shown in Fig. 4 - `Tuning fork`, `Low base tuning fork`, `Bridge`, `Arch`, `Snake`, `Horse`, `Frame` and `Stacking`. Please see Baskaran et al. [21] for more details on the dataset collection procedure. We manually annotated the dataset instances to add labels and post-processed them to get the following features per instance: Task (T), Recorded actions ($A_1, A_2, ..., A_n$), Images of the workspace ($I_0, I_1, ..., I_n$), States of the workspace ($S_0, S_1, ..., S_n$) and Possible tasks ($Tp_0, Tp_1, ..., Tp_n$). T denotes the goal structure. A recorded action A_i is of the format `<Left robot action (`A_{il}`), Right robot action (`A_{ir}`)>`. A_{il} and A_{ir} can be one of the nine actions - `Pick block` B_x, `Stand block` B_x, `Lie block` B_x, `Side-lie block` B_x, `Stand-on-block block` B_x, `Lie-on-block block` B_x, `Side-lie-on-block block` B_x, `Idle`, and `Withdraw`. An image I_i is taken from the video where A_i ends and A_{i+1} begins. We also added block numbers to the images in post-processing to distinguish the blocks (See Fig. 2). S_i is the structured state representation of I_i. Tp_i refers to all possible structures that share S_i as an intermediate state, as some tasks may overlap with others. For example, the Arch task overlaps with the Bridge and Horse tasks (see Fig. 4). For our evaluation we used the initial part of the recorded actions ($A_1, A_2, ..., A_k$) and the image of the workstation after completing those actions (I_k), to predict the remaining actions required to complete the task (T).

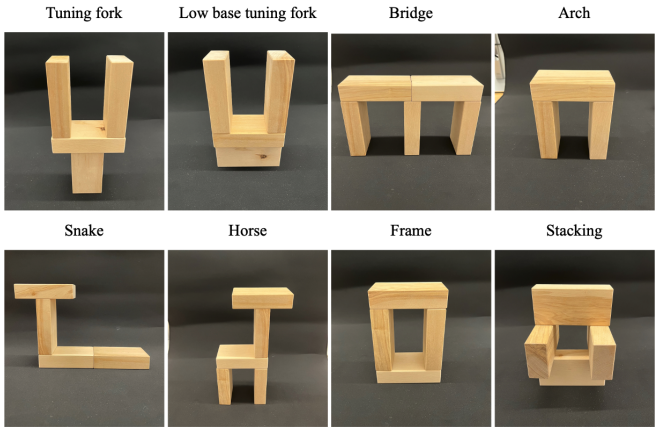

Fig. 4: The eight block assembly tasks.

B. Metrics

To calculate quantitative metrics for the predicted actions, we first manually propagate the ground-truth current state of the workstation using hand-crafted rules. Our rules take as input the predicted actions, and the predicted states which help to determine assistance parameters to remove ambiguity from action execution. The baseline (See §IV) does not predict future states and thus suffers from ambiguity in execution. So we randomly select a state from the set of possible states that can result from executing the actions predicted by the baseline. Note that here we select a state only from those reachable by the predicted actions, not from all possible states. Following existing research [18], [19], we show results across three metrics: executability, final-state error, and longest common subsequence. We also provide the task correctness and current state error to measure the correctness in predicting the overall task and the error in inferring the current state of the workspace.

Executability ensures that the predicted actions follow a logical order and satisfy execution constraints (e.g., proper object poses, action preconditions, etc.). We perform checks for executability and truncate the predicted action sequence at the first action that fails the checks. We report executability as the percentage of predicted actions that passed all the checks.

Final-state error measures the error of the ground truth task completion state from the final state that could be successfully achieved by executing the predicted actions. For the final achievable state, we select the resulting state from the last predicted action that can be executed successfully. This metric measures the difference between the result and the expected goal.

Longest Common Sub-sequence (LCS) is defined as the length of the longest common sub-sequence of actions in the predicted actions and the remaining recorded actions from our dataset, divided by the length of the longer sequence between predicted actions and remaining recorded actions. Following prior work [18], [19], [48], we allowed gaps to exist between the actions in the common sub-sequence, provided their order was the same. We report LCS as a percentage. A high LCS can mean that the predicted actions are relevant to the task and have short-term order. However, it does not provide a complete picture of correctness on its own, as there can be multiple correct action sequences with varying LCS values. A high LCS also does not guarantee that the predicted actions will execute successfully.

Task correctness measures the accuracy of *task estimation* (See §III-A). It is assigned 1 if the predicted task matches any of the possible tasks at inference time; otherwise, it is set to 0.

Current state error compares the distance between the current state output of *state representation generation* (See §III-B) and the ground-truth state at the time of inference.

V. Results & Discussion

We used the GPT-4o [49] model offered by the OpenAI API [50] for our main results. Every time we prompt the

TABLE I: Comparison of executability, final error, and LCS of STEP with the baseline, along with task correctness and current state error for multiple task completion percentages.

Task Comp. %	Exec % ↑		Final Error ↓		LCS % ↑		Task Corr. % ↑	Curr. St. Err. ↓
	Base	STEP	Base	STEP	Base	STEP		
30	64.25	**85.29**	5.96	**5.08**	**49.62**	45.37	74.14	2.48
50	65.51	**84.55**	4.79	**4.05**	**44.46**	38.82	69.70	3.11
70	73.44	**85.30**	3.23	**2.64**	**38.01**	33.88	67.47	3.99
90	81.34	**82.38**	1.41	**0.91**	**20.00**	15.66	75.15	4.87

MM-LLM, we sample 5 outputs and pick the output with the maximum log-probability returned by the model.

A. Comparison with Baseline

To model variations in user delegations to STEP, we conduct experiments at multiple task completion percentages of 30%, 50%, 70%, and 90%. A task completion of $x\%$ signifies that the initial $x\%$ of the actions from the recorded action sequence were used as input for evaluation.

As can be seen in Table I, STEP predicted action plans that were more executable than the baseline, for all percentages of task completion. Also, by predicting future states along with actions, our method resulted in a state that was closer to the intended goal of the operator, when compared to the baseline. However, we noted that the baseline method had a higher LCS than our method. This behavior can be attributed to STEP truncating the predicted actions after the state with the least distance from the goal. As a result, our predicted action sequence was on average approximately 3 actions shorter in length than the baseline's predictions. We suspect that the extra actions produced by the baseline caused it to have more number of actions in common with the recorded actions, thus increasing the baseline LCS. However, the higher LCS of the baseline resulted in unnecessary actions that were less executable and less correct in the resultant state. Action plans generated by STEP were more efficient, as they had a lower final error compared to the baseline, while being shorter on average. We hypothesize that noise in the recorded actions, caused by operator errors when collecting the data also contributed to LCS being a questionable metric in our evaluation.

We noticed that with increasing task completion percentage, both the methods achieved a better final error, since the robot started from a state closer to the goal state, as more of the assembly was completed by the operator. With increasing task completion, the task correctness first declines due to increasing noise in longer input action sequences and more occlusion of the block structures in the images, and then increases as the structure being built becomes more discernible. We suspect that the current state error increased as the task progressed because the structure became more complicated and blocks got closer together in the image.

B. Ablations

We ablated the number of rollouts conducted before calculating the metrics. Table II shows the effect of the number of rollouts on performance for 50% task completion. We noticed that each rollout predicted a longer action sequence than the preceding rollout, with action plan length increasing by 0.7 actions on average with each rollout. LCS increased with more rollouts, which is explained by the increased number of actions in the action plan. The increasing actions did cause the executability to decrease. However, as seen from the decreasing final error with more rollouts, the added actions that could execute were necessary as they were able to result in a final state closer to the goal with increasing rollouts.

TABLE II: Ablation of rollouts for 50% task completion. Final error improves with rollouts showing that additional rollouts decrease the distance from goal.

Rollouts	Exec. % ↑	Final Err. ↓	LCS % ↑
1	**88.98**	4.40	31.06
2	85.60	4.11	37.4
3	84.55	**4.05**	**38.82**

TABLE III: Ablating MM-LLM to compare executability and final error of STEP with the baseline for 50% task completion. Results for 45 random dataset instances.

Model	Exec. % ↑		Final Error ↓		Task Corr. % ↑
	Base	STEP	Base	STEP	
GPT-4o	71.68	**78.12**	4.53	**4.38**	62.22
GPT-4o mini	**82.60**	82.16	**4.49**	4.93	26.67
GPT-4.1	82.28	**92.02**	4.80	**4.11**	71.11
GPT-5.1	69.24	**82.70**	4.80	**3.71**	73.33
GPT-5.2	76.64	**84.44**	4.73	**4.49**	71.11

We compared STEP with the baseline for various MM-LLMs - GPT-4o, GPT-4o mini, GPT-4.1, GPT-5.1, and GPT-5.2, offered by the OpenAI API [50]. These models were selected to cover a range of model sizes and generation paradigms (regular LLMs and reasoning LLMs). To reduce computational cost and runtime, this ablation was conducted on a representative subset of 45 dataset instances. These instances were randomly sampled from our dataset while ensuring coverage across all eight task categories (see Fig. 4). As seen in Table III, STEP outperformed the baseline across most models in terms of both executability and final error, demonstrating its robustness and generalizability across different MM-LLMs. For GPT-4o mini, however, the baseline achieved slightly better results than STEP. We observed that GPT-4o mini exhibited a notably low task correctness score,

which measures the task estimation stage, the first step in our pipeline, and consequently propagated errors throughout subsequent stages for both methods. As a result, the outputs corresponded to an incorrect task, rendering these results less reliable compared to those from other models. We attribute this behavior to GPT-4o mini's relatively smaller model size, which likely limits its task estimation capabilities.

TABLE IV: Ablation of task correctness for 50% task completion. All metrics improve with correct task prediction. Improvement in earlier stages improves downstream performance.

Task corr.	Exec. % ↑	Final Err. ↓	LCS % ↑
1	**84.63**	**3.64**	**41.16**
0	84.37	4.9	33.45

TABLE V: Ablation of current state error for 50% task completion. Executability and final error improve with no error in the current state estimation.

Curr. st. err.	Exec. % ↑	Final Err. ↓	LCS % ↑
0	**86.38**	**3.93**	36.08
>0	84.41	4.06	**39.07**

We also studied the impact of task and state estimation on action prediction and state propagation. Table IV compares the results for the data instances that had a correct task prediction with the instances with an incorrect task prediction. Similarly, Table V shows the results for instances that did not have any errors in their current state prediction compared to the other instances. On average, we saw an improvement in performance of the predicted actions if the task and current state were predicted correctly. This ablation also shows the modularity of our approach as stages can be worked on independently and an improved performance earlier in the pipeline can reliably be transferred to better results for the final predicted actions.

VI. Conclusion

We introduce STEP, a method to enhance long-term action anticipation with MM-LLMs by explicitly modeling state transitions, leading to more accurate and executable robot action plans in collaborative industrial tasks. We construct a multi-staged pipeline composed of first estimating the overall task a robot operator is trying to perform, along with a structured representation of the current state of the robot's workstation. We then pass this information on to later stages to predict the actions required to complete the task and propagate the state representation to obtain the assistance parameters required to successfully execute the predicted actions for the desired task. We also track the distance between the propagated states and the goal state and guide predicted actions towards the goal.

We extensively evaluated our method on a dataset of an assembly scenario with teleoperated robots. Our results highlight our contribution towards bridging some of the gaps in the existing literature by demonstrating improved performance in executability and final error over the existing state-of-the-art. However, we see some areas for improvement. One limitation of STEP is that its current implementation is specific to the block assembly scenario. Industry may extend STEP to other scenarios by using information relevant to the new scenarios, by modifying the prompts, state representation structure, and function implementations. This investment in modifying STEP for a new scenario would only be required once at the start. However, future work can explore ways to generalize STEP to various scenarios with minimal modifications. A potential avenue of research could be using an MM-LLM to analyze a scenario and automatically craft the required prompts, state representation structure, and function implementations.

Additional work is also required to make STEP real-time before it can be adopted for practical use. In order to propagate states and guide the generation of action plans towards the goal, STEP queries an MM-LLM several more times than the baseline. However, this causes STEP to trade off speed for performance accuracy. In our experiments, we observed that the baseline had a shorter run time (M = 18.24, SD = 4.23 sec) than STEP (M = 148.07, SD = 55.03 sec). Future research could investigate hosting an MM-LLM locally or finetuning a smaller model to reduce the latency of each query. We hope that our work encourages further inquiries into the potential of MM-LLMs in long-horizon reasoning for human-robot collaboration.

References

[1] A. Pichler, S. C. Akkaladevi, M. Ikeda, M. Hofmann, M. Plasch, C. Wögerer, and G. Fritz, "Towards shared autonomy for robotic tasks in manufacturing," *Procedia Manufacturing*, vol. 11, pp. 72–82, 2017.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[3] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[5] F. Sener, D. Singhania, and A. Yao, "Temporal aggregate representations for long-range video understanding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*. Springer, 2020, pp. 154–171.

[6] Y. Abu Farha, A. Richard, and J. Gall, "When will you do what?-anticipating temporal occurrences of activities," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5343–5352.

[7] Y. Abu Farha, Q. Ke, B. Schiele, and J. Gall, "Long-term anticipation of activities with cycle consistency," in *Pattern Recognition: 42nd DAGM German Conference, DAGM GCPR 2020, Tübingen, Germany, September 28–October 1, 2020, Proceedings 42*. Springer, 2021, pp. 159–173.

[8] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes, "Forecasting future action sequences with neural memory networks," *arXiv preprint arXiv:1909.09278*, 2019.

[9] Q. Zhao, S. Wang, C. Zhang, C. Fu, M. Q. Do, N. Agarwal, K. Lee, and C. Sun, "AntGPT: Can large language models help long-term action anticipation from videos?" in *The Twelfth International*

Conference on Learning Representations, 2024. [Online]. Available: https://openreview.net/forum?id=Bb21JPnhhr

[10] E. M. Bender and A. Koller, "Climbing towards nlu: On meaning, form, and understanding in the age of data," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 5185–5198.

[11] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[12] S. Hao, Y. Gu, H. Ma, J. J. Hong, Z. Wang, D. Z. Wang, and Z. Hu, "Reasoning with language model is planning with world model," *arXiv preprint arXiv:2305.14992*, 2023.

[13] Z. Liu, H. Hu, S. Zhang, H. Guo, S. Ke, B. Liu, and Z. Wang, "Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency," *arXiv preprint arXiv:2309.17382*, 2023.

[14] S. Chen, A. Xiao, and D. Hsu, "Llm-state: Open world state representation for long-horizon task planning with large language model," *arXiv preprint arXiv:2311.17406*, 2024.

[15] K. Xie, I. Yang, J. Gunerli, and M. Riedl, "Making large language models into world models with precondition and effect knowledge," *arXiv preprint arXiv:2409.12278*, 2024.

[16] T. Yoneda, J. Fang, P. Li, H. Zhang, T. Jiang, S. Lin, B. Picker, D. Yunis, H. Mei, and M. R. Walter, "Statler: State-maintaining language models for embodied reasoning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 15 083–15 091.

[17] S. Zhou, T. Zhou, Y. Yang, G. Long, D. Ye, J. Jiang, and C. Zhang, "Wall-e: World alignment by rule learning improves world model-based llm agents," *arXiv preprint arXiv:2410.07484*, 2024.

[18] M. Gramopadhye and D. Szafir, "Generating executable action plans with environmentally-aware language models," 2023.

[19] W. Huang, P. Abbeel, D. Pathak, and 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.

[20] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and 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. 11 523–11 530.

[21] P. Baskaran, X. Liu, S. Li, and S. Iba, "eXplainable intention estimation in teleoperated manipulation using deep dynamic graph neural networks," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025.

[22] G. Li, Q. Li, C. Yang, Y. Su, Z. Yuan, and X. Wu, "The classification and new trends of shared control strategies in telerobotic systems: A survey," *IEEE Transactions on Haptics*, vol. 16, no. 2, pp. 118–133, 2023.

[23] M. Selvaggio, M. Cognetti, S. Nikolaidis, S. Ivaldi, and B. Siciliano, "Autonomy in physical human-robot interaction: A brief survey," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7989–7996, 2021.

[24] M. Cai, K. Patel, S. Iba, and S. Li, "Hierarchical deep learning for intention estimation of teleoperation manipulation in assembly tasks," *arXiv preprint arXiv:2403.19770*, 2024.

[25] W. Yu, R. Alqasemi, R. Dubey, and N. Pernalete, "Telemanipulation assistance based on motion intention recognition," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 1121–1126.

[26] M. Gao, J. Oberländer, T. Schamm, and J. M. Zöllner, "Contextual task-aware shared autonomy for assistive mobile robot teleoperation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3311–3318.

[27] M. Li and A. M. Okamura, "Recognition of operator motions for real-time assistance using virtual fixtures," in *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings.* IEEE, 2003, pp. 125–131.

[28] D. Aarno and D. Kragic, "Motion intention recognition in robot assisted applications," *Robotics and Autonomous Systems*, vol. 56, no. 8, pp. 692–705, 2008.

[29] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization for teleoperation and teaming," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 717–742, 2018.

[30] T.-C. Lin, A. U. Krishnan, and Z. Li, "Shared autonomous interface for reducing physical effort in robot teleoperation via human motion mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9157–9163.

[31] S. Manschitz and D. Ruiken, "Shared autonomy for intuitive teleoperation," in *ICRA Workshop: Shared Autonomy in Physical Human-Robot Interaction: Adaptability and Trust*, 2022.

[32] D. Zhang, R. Tron, and R. P. Khurshid, "Haptic feedback improves human-robot agreement and user satisfaction in shared-autonomy teleoperation," in *2021 ieee international conference on robotics and automation (icra)*. IEEE, 2021, pp. 3306–3312.

[33] C. Wang, S. Hasler, D. Tanneberg, F. Ocker, F. Joublin, A. Ceravola, J. Deigmoeller, and M. Gienger, "Lami: Large language models for multi-modal human-robot interaction," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–10.

[34] J. Davison, J. Feldman, and A. Rush, "Commonsense knowledge mining from pretrained models," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1173–1178. [Online]. Available: https://aclanthology.org/D19-1109

[35] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" *CoRR*, vol. abs/1911.12543, 2019. [Online]. Available: http://arxiv.org/abs/1911.12543

[36] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language Models as Knowledge Bases?" *ArXiv*, vol. abs/1909.01066, 2019.

[37] G. Ilharco, R. Zellers, A. Farhadi, and H. Hajishirzi, "Probing Text Models for Common Ground with Visual Representations," *ArXiv*, vol. abs/2005.00619, 2020.

[38] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training Verifiers to Solve Math Word Problems," *ArXiv*, vol. abs/2110.14168, 2021.

[39] J. Shen, Y. Yin, L. Li, L. Shang, X. Jiang, M. Zhang, and Q. Liu, "Generate & rank: A multi-task framework for math word problems," *arXiv preprint arXiv:2109.03034*, 2021.

[40] K. Lu, A. Grover, P. Abbeel, and I. Mordatch, "Pretrained Transformers as Universal Computation Engines," *arXiv preprint arXiv:2103.05247*, 2021.

[41] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[42] M. Tsimpoukelli, J. Menick, S. Cabi, S. Eslami, O. Vinyals, and F. Hill, "Multimodal few-shot learning with frozen language models," *Proc. Neural Information Processing Systems*, 2021.

[43] M. M. Islam, T. Nagarajan, H. Wang, F.-J. Chu, K. Kitani, G. Bertasius, and X. Yang, "Propose, assess, search: Harnessing llms for goal-oriented planning in instructional videos," in *European Conference on Computer Vision*. Springer, 2024, pp. 436–452.

[44] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[45] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo, "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *arXiv preprint arXiv:2305.15021*, 2023.

[46] H. Corporation, "Htc vive pro controllers," https://www.vive.com/, 2025.

[47] F. E. GmbH., "Franka emika robot: Instruction handbook," 2021.

[48] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8494–8502.

[49] OpenAI, "Hello GPT-4o," 2024. [Online]. Available: https://openai.com/index/hello-gpt-4o/

[50] G. Brockman, P. Welinder, M. Murati, and OpenAI, "Openai: Openai api," 2020. [Online]. Available: https://openai.com/blog/openai-api