

# Multi-Agent Systems for Robotic Autonomy with LLMs

Junhong Chen, Ziqi Yang, Haoyuan G Xu, Dandan Zhang, George Mylonas\*  
Imperial College London

{junhong.chen16, z.yang21, g.xu23, d.zhang17, george.mylonas}@imperial.ac.uk

## Abstract

*Since the advent of Large Language Models (LLMs), various research based on such models have maintained significant academic attention and impact, especially in AI and robotics. In this paper, we propose a multi-agent framework with LLMs to construct an integrated system for robotic task analysis, mechanical design, and path generation. The framework includes three core agents: Task Analyst, Robot Designer, and Reinforcement Learning Designer. Outputs are formatted as multimodal results, such as code files or technical reports, for stronger understandability and usability. To evaluate generalizability comparatively, we conducted experiments with models from both GPT and DeepSeek. Results demonstrate that the proposed system can design feasible robots with control strategies when appropriate task inputs are provided, exhibiting substantial potential for enhancing the efficiency and accessibility of robotic system development in research and industrial applications.*

## 1. Introduction

In recent years, Large Language Models (LLMs) like OpenAI's GPT, Meta's LLaMA, and Anthropic's Claude, have emerged as powerful tools, enabling advanced language understanding, reasoning, and problem-solving [1–3]. They were initially developed for Natural Language Processing (NLP) but now have been integrated into various fields. Particularly in robotics, they have significant potential for promoting robotic autonomy by enhancing perception, control, and decision-making [4–6]. They can translate natural language descriptions into executable robotic actions, reducing the requirements for task-specific programming. Additionally, the rapid growth of LLM-based multi-agent systems (MAS) has demonstrated a strong potential for collaborative tasks by introducing communication mechanisms among multiple agents, which allow robots to share environmental information, adapt strategies, and efficiently work together toward a common goal [7–9].

Reinforcement Learning (RL) has witnessed significant advancements in recent years because of its iterative

human-like approach to learning and decision-making. Disparate from traditional data-driven methods, RL typically requires minimal prior knowledge, as agents learn by actively interacting with the environment and receiving feedback through rewards or penalties [10, 11]. Such architecture is particularly feasible for sequential decision-making problems in robotics. RL empowers robots to acquire complex behaviors through trial and error, progressively enhancing their performance and allowing more effective adaptability in dynamic or unstructured environments [12, 13]. RL has been successfully applied to tasks such as path planning, robot control, object manipulation, and multi-agent coordination, which all represent critical components of achieving robotic autonomy [14, 15].

Attributed to the powerful language comprehension and execution efficiency of LLMs, their combination could enhance robot learning by integrating language reasoning and decision-making abilities. LLMs can act as advanced planners, converting natural language descriptions into structured, machine-readable commands that guide RL agents to focus on crucial decisions and accelerate policy learning in complex environments or those real cases which lack of control strategies [16–18]. Furthermore, multi-agent communication architectures enable the decomposition of tasks into sub-tasks that can be executed efficiently, either in parallel or sequentially, simplifying control and enhancing efficiency. For instance, by incorporating multi-agent RL, robotic swarms can interpret commands, generate strategies, and dynamically adapt to complex environments based on language-driven patterns and task-activated reasoning from LLMs [19, 20].

In this study, we propose a multi-agent, LLM-based framework to support robotic task analysis, mechanical design, and path generation. The main contributions include:

- Introduce a modular LLMs-based framework for flexible and scalable robotic development, featuring three core agents for robotic task analysis, robot design, and reinforcement learner training, with sub-agents for extracting codes and summarizing reports from outputs.
- Develop human-friendly methods for generating readable and reliable technical analysis reports, path visualiza-

\*Corresponding author

- tions, and RL codes, ensuring clarity, practicality, and actionable results for robotic system design.
- Conduct experiments with several LLMs in different performance levels to validate the generalizability and efficiency of the proposed framework, and ablation studies to clarify the functions and impacts of each agent.

## 2. Related work

This section will review the application of LLMs and MAS in the field of robotic autonomy, and relevant research on the reinforcement learning based robotic path generation.

### 2.1. LLMs in Robotics

The intelligence required for robotics research has been steadily increasing. Traditional approaches have relied on functional modules with neural network algorithms, some incorporating vision algorithms such as NLP and YOLO to provide a certain degree of intelligence on communication and image reading receptively [21–24]. However, with the widespread adoption of LLMs, robotics research is no longer limited to functionally independent systems but shifts to task-oriented frameworks [25, 26]. Although there is no formal mathematical proof that these models enhance robotic intelligence, recent advancements in task-oriented research based on the capacities of LLMs indeed reveal a promising trend [27, 28].

For instance, Google introduced PaLM-E in 2022, a large model specifically designed for robotic task training [29]. More recently, LLMs with reasoning capabilities, such as GPT-O1 and DeepSeek-R1, have significantly addressed the challenges of autonomous logical inference in robotics [30, 31]. LLMs have made significant improvements in various subfields, including autonomous task sequence generation based on task descriptions, detailed task refinement, multi-robot collaboration, and adaptive human-robot interaction [32–37].

### 2.2. Multi-Agent Systems in Robotics

Unlike the single robotic module, MAS offers a more human-like and flexible approach to global robotic system design. In MAS-based robotics, each functional module is assigned to a distinct agent, facilitating information processing of different levels and agent-agent communication. By employing a multi-agent framework, robotic systems now walking into a future from a step over single-function demonstrations to fully autonomous systems capable of decision-making, task planning, and execution.

For instance, robots guided by high-level instructions can autonomously determine task execution strategies. The RoCo framework, developed by Mandi, assigns individual robotic agents to each robotic arm in a multi-arm system, enabling coordination through agent interaction [35]. Similarly, multi-agent robots are applicable in multi-robot collaboration systems, where inter-agent communication enhances efficiency and flexibility [38]. MAS improves the

adaptability of robotic applications across various subfields and allows them to be seamlessly blended.

## 2.3. Reinforcement Learning for Robotic Control

Reinforcement learning (RL) has become an essential tool in robotics, allowing systems to adapt to diverse task environments and motion constraints. Various RL algorithms are employed based on task-specific requirements. The most commonly used reinforcement learning algorithms include:

**Q-Learning and DQN (Deep Q-Learning):** Suitable for tasks requiring discrete iterative motion, such as path finding and obstacle avoidance [39, 40].

**A3C (Asynchronous Advantage Actor-Critic):** Enhances stability and efficiency through asynchronous updates and advantage functions, making it ideal for object grasping [41].

**PPO (Proximal Policy Optimization) and TRPO (Trust Region Policy Optimization):** Improve policy optimization techniques, making training for navigation and robotic arm control more stable and efficient [42–44].

**DDPG (Deep Deterministic Policy Gradient) and SAC (Soft Actor Critic):** Aim for continuous motion control; SAC further incorporates entropy regularization to enhance exploration capabilities [15, 45].

**Multi-Agent RL:** Algorithms such as Independent Q-Learning (IQL) and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) enable independent reinforcement learning for each agent, enhancing cooperative intelligence in MAS of robotics [46].

## 2.4. Path Generation

Path generation is a fundamental problem in robotic kinematics, as it determines how a robot moves at each timestep to complete a given task or reach a target position. Various methods are utilized for trajectory planning:

**Model Predictive Control (MPC):** Employs control-based optimization to generate efficient trajectories [47, 48].

**Imitation Learning:** Allows robots to mimic expert demonstrations to complete similar tasks [49].

**Reinforcement Learning:** Emphasizes autonomous exploration of feasible trajectories based on environmental feedback, enabling robots to learn and generate paths from nothing [15, 40].

**Few-Shot and Zero-Shot Learning:** Emerging research focuses on accelerating robot learning for simple tasks with minimal training samples [50].

**Data-driven Deep Learning:** Enables the robot to generate end-effector trajectories based on sparse constraints or limited task-specific conditions [51, 52].

These methods, with the advent of LLMs, are showing potential outcomes of robotic task autonomy, enabling robots to perform tasks with less human intervention.

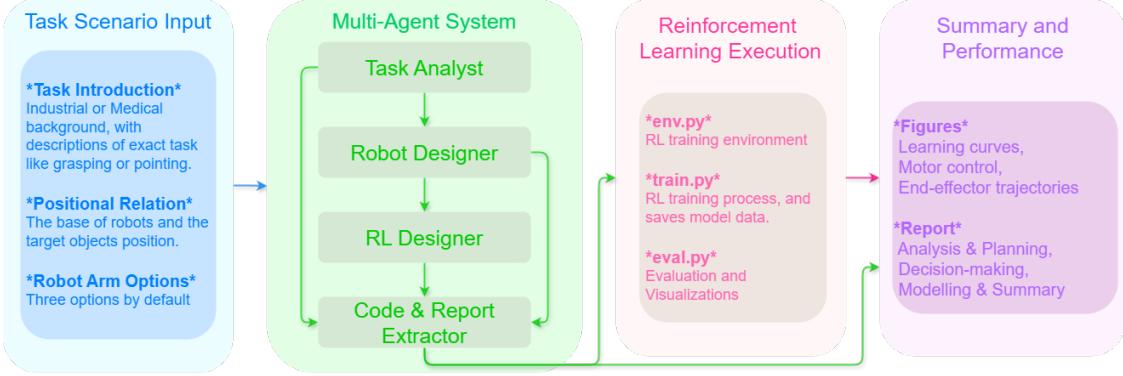


Figure 1. The overview of the proposed multi-agent robot system

### 3. Method

#### 3.1. Overview of the framework

An overview of the proposed multi-agent robot system framework, is illustrated in Fig.1. The system takes task scenario descriptions as the only input. To ensure system stability, the input must clearly define the following three aspects: 1) the operational environment of the task, 2) the configuration of robot base options and the target positions, and 3) the available robotic arm length options.

Once the task descriptions are clearly specified, the first agent, the Task Analyst, receives the input and performs an engineering-oriented analysis. This agent extracts key information from the task description, establishes a coordinate system based on the provided robot base and target positions, and converts all relevant positional data into the coordinate frame. However, the specification of robotic arm length options is kept without modification. The final output of this agent is a Task Analysis Report, consisting of both extracted and preserved task information.

The second agent, the Robot Designer, processes the Task Analysis Report and extracts necessary details, such as robot base coordinates, target positions, and available arm length options. This agent is responsible for analyzing the task allocation strategy within the given environment, determining the required number of robots, assigning sub-tasks to each robot, and selecting appropriate robotic arms with varying length options. The design process considers both economic efficiency and operational safety by its understanding of the background of task scenarios, ensuring that the selected arms are neither excessively long nor insufficiently short. The final Robotic Design Report is then generated and passed to the next agent.

The third agent, the Reinforcement Learning Designer, utilizes the Robotic Design Report to generate all necessary RL components. This agent is also required to address RL model selection and implementation issues. The final output includes a comprehensive RL implementation report, consisting of an analysis of the RL framework and three key code modules: Environment Definition, Training Script and

Evaluation Script, which will be detailed introduced.

Once completing all agent tasks, the reinforcement learning module executes the generated code to train the initial task model. The system then outputs figures and a final report.

#### 3.2. Multi Agents Integration

This section provides a detailed introduction of the multi-agent system architecture, as depicted in Fig.2. The system consists of three core agents and two additional agents, with the core agents arranged in a linear workflow, while the others dealing with their report for final outputs.

##### 3.2.1. Task analyst

The Task Analyst processes the task scenario description as input and performs a structured engineering analysis. The key prompts include:

1. Determining the number of robots required and assigning base positions based on the task description,
2. Identifying the target points and their coordinates,
3. Summarizing and relaying other task-specific requirements.

The Task Analysis Report generated by this agent consists of five key elements:

1. Number of Targets to be Reached, 2. Number of Robots to be Built, 3. Base Location Options, 4. Arm Link Length Options, 5. Arm Choices Information.

This report is then forwarded to the Robot Designer for further processing.

##### 3.2.2. Robot designer

The Robot Designer refines the preliminary engineering analysis into a modeling-ready decision report. The main tasks of this agent include:

1. Extracting key details from the Task Analysis Report to generate a comprehensive system-level plan.
2. Selecting suitable base locations for robots and allocating sub-tasks accordingly.
3. Determining optimal robotic arm configurations, ensuring that each robot can effectively reach all assigned



Figure 2. The detailed design of LLMs-based Multi-Agent Systems in the Framework

target points while maintaining economic and redundancy considerations.

4. Summarizing all design choices before passing the information to the next agent.

The Robotic System Design Report contains five essential sections: 1. Required Number of Robots, 2. Selected Base Location, 3. Design Decisions for Robotic Arms, 4.

Final Robotic Arm Configuration, 5. Summary

Similarly, this report will be the input that is sent to the third core agent: the RL Designer.

### 3.2.3. RL designer

Since no human demonstrations and prior knowledge is provided, the RL Designer is the most crucial agent of this MAS framework, as it transforms task requirements into

an operational reinforcement learning model and generates trajectories for task requirements. However, it deeply relies on the information processed from the Task Analyst and Robot Designer to ensure a well-structured foundation for learning-based training. This agent has two key aspects:

**RL Model Selection and Design:** The RL framework is told to be flexible and capable of adaptation. Based on the task environment and objects, the agent needs to select an appropriate RL algorithm and list its reasons behind.

**RL Code Implementation:** The agent generates RL-related code, defining the environment, motion policies, and success criteria.

The RL Design Report consists of five sections:

1. Environment Design, 2. Motor Motion Definition, 3. Reinforcement Learning Algorithm Selection, 4. Success and Failure Criteria, 5. Initial Conditions

Additionally, three independent code files are generated:

**Environment Definition (env.py):** Defines the RL training environment and its initialization, resetting and interaction functions.

**Training Script (train.py):** Runs the RL training process, and saves model data.

**Evaluation Script (eval.py):** Executes the trained model with given initials and visualizes control data and end-effector trajectories.

### 3.2.4. Code and Report extractor

**Code Extractor:** Since agents cannot directly execute the generated code, a Code Extractor is necessary to extract all code components mentioned in the reports and divides them into separate files for execution(env.py, train.py, and eval.py).

**Report Extractor:** Merges all output reports into a final comprehensive report, summarizing the entire intelligent analysis, decision-making, execution strategies, modeling, but excluding code.

## 3.3. Reinforcement Learning Execution

To keep identical simulation across all experiments, the RL simulations are conducted using OpenAI Gym as the standard environment. This ensures that all scenarios and results are evaluated within a unified benchmarking framework.

Key RL components, states, actions, and awards, are all determined by the RL Designer. This framework highlights the autonomous decision-making capabilities of the MAS, as the self-selected RL algorithms and RL components directly determine the training process and final outcomes.

## 3.4. Output and Evaluation

When executing the RL scripts (eval.py), the system generates the following outputs:

1. Learning curves from the RL training process.

2. Motor control visualizations, detailing the joint movements of each robot.

3. End-effector trajectories, showing the tip motion paths learned by the robotic arms.

The output figures include: the learning curves, motor control graphs, and robotic end-effector motion trajectories. The final report, generated by the Report Extractor, provides a structured summary of all intelligent analysis, planning, decision-making, modeling, and summary, illustrating the effectiveness of the proposed MAS for RL-powered robotic task autonomy.

The following section will conduct different experiments within this framework, to evaluate its generalizability and key influence of each core agents.

## 4. Experiments

### 4.1. Experimental Setup

Since this study involves four LLMs with varying levels of performance: GPT-4o-mini, DeepSeek-V3, GPT-4o, and DeepSeek-R1. To evaluate the performance of the proposed multi-agent system with different models, we conducted two types of experiments to assess whether these LLMs function well or not within the framework:

**Generalization Across Tasks:** This experiment evaluates how well when different AI models are adapted to different task descriptions within the MAS.

**Ablation Study:** This experiment quantifies the impact of three core agents in the MAS by disabling one or two agents separately to evaluate their output performance.

### 4.2. Generalization Across Tasks

To evaluate the generalization capability of both the models and the multi-agent framework, the paper designed ten task scenarios, divided into two categories: five industrial tasks and five medical tasks. The dual-scenario design may force the models to face different realistic case-specific challenges, leading to a generalizability check of task analysis of how different AI models perform. Each AI model is required to process all ten task descriptions and execute them using the proposed multi-agent system.

**Scenario Design:** Table.1 presents key details of the ten task scenarios. Recall that the input must contain:

Task title and description

Robot and target object positional information

Options for robotic arm lengths

To rigorously test the comprehension and summarization capabilities of the models, each task description consists of 100 to 150 words detailing the task requirements. Table.1 also provides an example task description at the bottom.

### 4.3. Ablation Study

To systematically quantify the role of each core agent within the framework and to evaluate the framework's sensitivity to the level of details of task descriptions, we conduct the ablation study from two perspectives:

Scenario	Task Name	Base Location Options	Target Location	Arm Options
1	Rehabilitation Therapy	(0,0) or (0.5,0)	(0.5,1.2), (0.8,1.5), (1.0,1.0)	0.8m, 1.0m, 1.2m
2	Surgical Instrument Handling	(0,0.5) or (0.2,0.3)	(0.5,0.5), (0.7,0.7), (1.0,0.6)	0.7m, 0.9m, 1.1m
3	Elderly Feeding Assistance	(0,-0.5) or (-0.3,-0.5)	(0.4,0.2), (0.5,0.5), (0.6,0.3)	0.6m, 0.8m, 1.0m
4	Physical Therapy Stretching	(0.5,0) or (0.3,-0.2)	(0.5,1.0), (0.6,1.2), (0.8,1.1)	0.9m, 1.1m, 1.3m
5	Prosthetic Limb Training	(0,0) or (0.2,-0.2)	(0.3,0.4), (0.5,0.6), (0.7,0.5)	0.7m, 0.9m, 1.2m
6	Assembly Line Placement	(0,0) or (0,0.3)	(0.4,0.3), (0.6,0.5), (0.8,0.4)	0.8m, 1.0m, 1.2m
7	Warehouse Item Sorting	(0,0) or (-0.5,0)	(0.5,1.0), (0.7,1.2), (1.0,1.1)	0.9m, 1.1m, 1.3m
8	Automobile Welding	(0,0) or (1.2,0.5)	(0.4,0.2), (0.6,0.3), (0.8,0.4)	0.7m, 0.9m, 1.0m
9	Pick-and-Place for Electronics	(0,0) or (0.2,0.3)	(0.3,0.4), (0.5,0.5), (0.7,0.6)	0.6m, 0.8m, 1.0m
10	Palletizing in Logistics	(0,0) or (0.5,0.5)	(0.4,0.5), (0.6,0.7), (0.8,1.0)	0.9m, 1.2m, 1.5m
Example	"There is a factory with two base for robot manipulators available and the gap of them are 10m, on the front, which is 20m of their middle point, there are 4 boxes need to be picked up, each of them has a 5m gap, and all boxes forms a line which is parallel to the robots base line. The robot could be designed from three following lengths of robot arms: 10m, 5m, and 2m, a robot could have multiple arms to form a serial robot."			

Table 1. key details of the ten task scenarios, Scenario 1-5 are industrial cases while Scenario 6-10 are medical cases. Example Description is used as the normal description for the ablation study.

One Agent Disable Ablation Condition	Two Agents Disable Ablation Condition
C1: Without Task Analyst (Core Agent 1)	C12: Without Task Analyst and Robot Designer (Core Agent 1&2)
C2: Without Robot Designer (Core Agent 2)	C13: Without Task Analyst and RL Designer (Core Agent 1&3)
C3: Without RL Designer (Core Agent 3)	C23: Without Robot Designer and RL Designer (Core Agent 2&3)

Table 2. Ablation configurations for proposed experiments.

#### 4.3.1. Robustness to Task Description

To investigate how the system responds to different detail levels of task description, we conduct ablation tests using three levels of task description length ranging from highly abstract instructions (e.g., "Pick up a box") to normal(as the example in Table.1), and to highly detailed specifications (e.g., "In a medical environment, grasping surgical tools requires extremely precise handling. For the current target, a medical-grade surgical instrument, previous studies typically utilize a three-joint robotic arm. The grasping process should approach the object at a stable yet relatively high speed before transitioning into a fine-tuned gripping maneuver.").

#### 4.3.2. Impact of Key Agents

To quantify the contribution of each core agent, this time we perform controlled experiments using a fixed task description (Example in Table.1). During this experiment, we systematically disable specific one or two agents while ensuring all other components function normally. Table.2 outlines the specific ablation configurations.

#### 4.4. Evaluation Metrics

To quantify those performances, we define the following evaluation metrics for both experiments, each metric has a score ranging from 0 to 5:

**Task Completion Progress (TCP):** Measures to what extent each task successfully and properly progresses through the multi-agent system. Due to the purpose of ablation studies, the agent may not finish all of those cases.

**Code Execution Feasibility (CEF):** Assesses whether the generated code runs successfully. LLMs may not always have correct and executable code. Some code will run smoothly after minor fixes, but some will generate code that does not run smoothly or use the correct libraries, function calls, and the correct version.

**Model Alignment (MA):** Determines if the final execution outcomes meet task requirements based on the generated code. While code could be run properly, the result of each case may not meet the initial requirements or even without convergence.

**Robot Design Adaptability (RDA):** Evaluates whether the robot design decisions align with task requirements. Robot design requires years of field operation and design experience, robot may not be designed well with all experimental cases.

**Report Maturity (RM):** Evaluates the clarity and completeness of the final output report, ensuring it provides a structured analysis of each step. The final report reflects the MAS' global cognition and detail processing ability for the given task description, and the level of this ability varies with different models or cases.

While TCP, CEF, and MA are objective metrics, the other two metrics, RDA and RM, are relatively subjective, since the primary purpose of the multi-agent system is to enhance intelligent decision-making. To ensure a convincing evaluation, four researchers with several years of research specializing in robotics and LLMs independently score these subjective metrics, and the average of the four scores for these two metrics is used as the final score.

Model	GPT-4o-mini					DeepSeek-V3					GPT-4o					DeepSeek-R1				
Scenario	TCP	CEF	MA	RDA	RM	TCP	CEF	MA	RDA	RM	TCP	CEF	MA	RDA	RM	TCP	CEF	MA	RDA	RM
1	5.00	2.00	1.00	3.25	3.00	5.00	2.00	0.00	2.75	3.25	5.00	3.00	0.00	4.50	4.00	5.00	4.50	4.50	3.50	4.50
2	2.00	3.00	0.00	2.50	3.00	5.00	3.00	1.00	3.75	3.00	5.00	3.50	1.00	4.00	3.25	2.00	4.00	2.00	3.50	3.75
3	5.00	4.00	2.00	3.50	3.25	5.00	4.00	2.00	4.25	4.25	2.00	4.00	2.00	3.00	4.00	5.00	5.00	3.00	4.00	3.75
4	2.00	4.00	2.00	3.00	3.75	2.00	3.00	0.00	3.00	3.75	2.00	4.00	2.00	3.75	3.50	5.00	5.00	5.00	4.25	4.00
5	4.00	3.00	0.00	3.50	3.25	2.00	3.50	2.00	3.50	3.75	4.00	4.00	3.50	3.75	4.00	5.00	4.00	5.00	4.00	4.50
6	2.00	5.00	1.00	3.25	4.25	2.00	2.00	1.00	3.50	3.25	2.00	3.00	1.00	3.25	4.25	5.00	4.00	3.00	3.75	4.00
7	4.00	0.00	0.00	4.00	4.00	4.00	4.50	1.00	3.50	3.50	5.00	4.00	1.00	4.25	3.50	4.00	4.50	3.00	4.00	3.75
8	2.00	4.50	2.00	3.50	4.00	2.00	2.00	1.00	3.50	4.25	2.00	3.00	2.00	3.75	3.25	5.00	4.00	5.00	3.50	4.25
9	2.00	3.00	1.00	3.50	3.75	4.00	4.00	2.00	3.00	3.50	4.00	3.00	1.00	3.75	3.25	5.00	5.00	3.00	4.00	4.50
10	2.00	5.00	1.00	4.00	4.00	2.00	2.00	1.00	3.00	3.50	5.00	4.00	2.00	3.25	3.50	5.00	4.50	5.00	4.25	4.50

Table 3. The score of five evaluation metrics across ten task descriptions. Metrics are: TCP-Task Completion Progress; CEF-Code Execution Feasibility; MA-Model Alignment; RDA-Robot Design Adaptability; RM-Report Maturity

Model	Metrics (Mean $\pm$ SD)				
	TCP	CEF	MA	RDA	RM
GPT-4o-mini	$3.00 \pm 1.33$	$3.35 \pm 1.53$	$1.00 \pm 0.82$	$3.40 \pm 0.44$	$3.63 \pm 0.46$
DeepSeek-V3	$3.30 \pm 1.42$	$3.00 \pm 0.97$	$1.10 \pm 0.74$	$3.38 \pm 0.44$	$3.60 \pm 0.41$
GPT-4o	$3.60 \pm 1.43$	$3.55 \pm 0.50$	$1.55 \pm 0.96$	$3.73 \pm 0.46$	$3.65 \pm 0.38$
DeepSeek-R1	$4.60 \pm 0.97$	$4.45 \pm 0.44$	$3.85 \pm 1.16$	$3.88 \pm 0.29$	$4.15 \pm 0.34$

Table 4. The Mean and SD values of scores of five evaluation metrics across ten task scenarios on each model.

## 5. Results

The experimental results for the experiments discussed in the previous sections are presented separately in the following two subsections. In addition, an example of output figures from code execution is shown in the last subsection.

### 5.1. Generalization Across Tasks

The score of evaluation metrics for generalization capability is shown in Table 3 and Table 4. Among the four models evaluated in this study, DeepSeek-R1 outperformed the others across all five evaluation metrics.

In TCP, one of the first three objective metrics, GPT-4o-mini and DeepSeek-V3 occasionally achieved high scores. However, their low scores were primarily attributed to two facts: 1) generate information that was not exist in the task requirements; 2) frequent decision-making errors, particularly in the robotic arm design. A common issue was their difficulty in correctly applying mathematical calculations, leading to systematic design errors. GPT-4o also encountered these problems, but less frequently, resulting in an overall better performance. DeepSeek-R1 performed the best due to its self-correction capabilities (reasoning model). This allowed it to verify and refine its decision-making process throughout task execution. Differing from other models, its occasional errors arose when making decisions between optimal and suboptimal solutions for robot design.

Regarding CEF, all models except DeepSeek-R1 received scores in the range of approximately 3. Common issues are due to inconsistent library and function calls,

variations in variable formats, and missing statements in code generation. While discrepancies in library versions and function calls can be addressed with minor adjustments, missing statements or wrong code structure usually result in bad outcomes, and show their limitations in generating executable code.

MA scores for the first three models were relatively low, causing their SD also looks relatively small. A key issue was that once the former analysis of the output report contained errors, then subsequent code execution deviated from the task requirements, the system completely got lost. Similarly, reinforcement learning procedures based on error design failed to converge, resulting in no meaningful results. Even DeepSeek-R1 occasionally came up with this issue. However, in most cases, it successfully generated the required output and ended in a convergent solution. An interesting phenomenon was that, among all 40 execution cases, DeepSeek-R1 was the only model that showed variability in selecting RL algorithms, between SAC and PPO. The other models always used PPO for these tasks.

For one of two subjective metrics, RDA, the overall scores of all models were relatively similar. This is likely due to the nature of LLMs in text generation. However, the reasons behind each models varied: GPT-4o-mini occasionally generated text that looking good at first glance, but finding errors when looking into details; DeepSeek-V3 exhibited inconsistencies in report quality; GPT-4o performed slightly better, with output quality close to DeepSeek-R1; DeepSeek-R1 did generate good reports, but sometimes it listed too much details that unable to find the key points.

Finally, in terms of RM, the first three models demonstrated relatively similar performance, while DeepSeek-R1 achieved a slightly higher score. This indicates that the DeepSeek-R1 may have better capability of generating proficient reports.

In summary, the approximate model performance level under this framework is as follows: GPT-4o-mini  $\approx$  DeepSeek-V3  $<$  GPT-4o  $<$  DeepSeek-R1.

Model	GPT-4o-mini					DeepSeek-V3					GPT-4o					DeepSeek-R1				
Scenario	TCP	CEF	MA	RDA	RM	TCP	CEF	MA	RDA	RM	TCP	CEF	MA	RDA	RM	TCP	CEF	MA	RDA	RM
Short	3.00	4.50	1.00	2.50	3.75	3.00	4.00	1.00	3.25	3.50	2.00	2.00	0.00	3.50	3.25	5.00	5.00	4.00	4.00	3.75
Normal	2.00	1.00	0.00	3.75	4.25	4.00	4.00	1.00	3.00	3.50	3.50	4.00	1.00	3.50	3.75	4.50	5.00	5.00	4.00	4.00
Long	4.00	1.00	0.00	3.00	3.50	3.00	3.00	0.00	3.25	4.00	2.00	4.00	1.00	3.00	3.75	5.00	5.00	3.00	3.50	3.50
C1	3.00	0.00	0.00	3.25	3.75	3.00	2.00	1.00	3.50	4.00	3.00	3.50	1.00	3.50	3.25	4.50	4.00	3.00	4.00	4.00
C2	2.00	1.00	0.00	2.50	3.50	3.00	3.50	2.00	2.50	3.75	3.00	3.50	1.00	3.00	3.75	4.50	4.00	4.00	3.50	4.00
C12	2.00	0.00	0.00	2.25	3.00	1.00	1.00	0.00	2.50	4.00	2.00	2.50	1.00	2.50	3.25	2.50	4.00	2.00	3.25	4.25
C3	2.00	-	-	2.75	3.75	4.00	-	-	3.50	4.25	4.50	-	-	3.50	3.50	4.50	-	-	4.00	4.75
C13	2.00	-	-	3.25	3.50	3.50	-	-	3.75	3.50	3.50	-	-	3.25	3.25	4.00	-	-	4.00	4.00
C23	1.00	-	-	2.25	2.75	2.00	-	-	3.25	4.25	1.00	-	-	3.50	3.75	4.00	-	-	4.50	4.50

Table 5. The score of five evaluation metrics for the ablation study. Metrics are: TCP-Task Completion Progress; CEF-Code Execution Feasibility; MA-Model Alignment; RDA-Robot Design Adaptability; RM-Report Maturity. Short, Normal, and Long indicate the task description length. C1, C2, etc. are the numbered ablation conditions same as Table.2

## 5.2. Ablation Study

In Table.5, for the first ablation study examining the impact of input task description length, the results indicate that more extensive task details do not correlatively enhance output performance in a MAS. The optimal description length leads to effective outputs in most cases, as reflected in the "Normal" column. We believed that excessively long input description lightens the weight of key information inside the output, while overly simple descriptions fail to provide sufficient contextual relevance, particularly for robot design and code generation.

It is likely that both 'Long' and 'Short' input introduce instability into the system. A proper input length enhances the creativity of the overall report while preventing task requirements from becoming biased due to insufficient information. Additionally, CEF and MA exhibit significant fluctuations in models other than DeepSeek-R1. While the code may still execute with minor modifications, decision-making in task analysis and reinforcement learning remains important. Additionally, DeepSeek-R1 also shows deviations in final outputs due to fluctuations in input description length. However, the two subjective metrics show similar values, ranging between 3 and 4. This may suggest that despite increased task difficulty, the model's outputs appear similar for the task report. The key differences lie in the process of transforming reports into actual designs, code, and results.

Regarding the second ablation study, in the absence of Core Agent 1, the generated results exhibit significant instability. Compared to the "Normal" column in the previous experiment, Task analyst plays a crucial role in defining the overall task direction. Without its guidance, all subsequent generation processes become ambiguous. Even DeepSeek-R1 experiences a decline in performance metrics when losing a high-level analytical summary. In the experimental comparison of Core Agent 2, Robot Designer, the system's information processing stopped due to its absence. RL designer struggles to obtain sufficient data for modeling and

learning, as meaningful results become difficult to generate. While DeepSeek-R1 shows an advantage by autonomously supplementing missing information, its accuracy and relevance remain uncertain. An interesting observation is that for GPT-4o-mini, the absence of Robot Designer leads to some blank space in reinforcement learning code generation. This causes its code to be entirely non-executable without manual intervention. At last, the impact of Core Agent 3, RL designer, is obvious, no reinforcement learning or control logic is present without RL Designer, and the tasks remain confined solely to the robot design stage.

## 6. Conclusion

In this paper, we proposed a MAS framework for robotic autonomy with LLMs. The proposed multi-agent framework can effectively design feasible robot configurations and produce corresponding control solutions when the task requirements are provided through natural language prompts with proper details. For LLMs with knowledge and capabilities at different levels, lower ones, especially GPT-4o-mini, and DeepSeek-v3, the robot design outcomes are not always feasible, resulting in wrong RL design while GPT-4o meets the borderline of requirements. The reasoning model DeepSeek-R1 keeps a higher quality output with quite a few mistakes. This approach exhibits promising potential for enhancing the efficiency and accessibility of robotic system development based on the knowledge of LLMs instead of human-only strategies, offering valuable insights for future advancements in intelligent robotics and industrial applications. Future work could focus on the following aspects: introducing obstacles and dynamic objects in the task scenarios; extending the proposed system into a multi-level, nested hierarchical interactive agent architecture; integrating large vision-language models (LvLMs) to extract spatial information from real-world scenarios; and deploying a voice module for continuous human-machine natural language communication.

## References

- [1] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. [1](#)
- [2] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. [1](#)
- [4] Liam Roy, Elizabeth A Croft, Alex Ramirez, and Dana Kulić. Gpt-driven gestures: Leveraging large language models to generate expressive robot motion for enhanced human–robot interaction. *IEEE Robotics and Automation Letters*, 2025. [1](#)
- [5] Yuzhi Lai, Shenghai Yuan, Youssef Nassar, Mingyu Fan, Atmaraaj Gopal, Arihiro Yorita, Naoyuki Kubota, and Matthias Rätsch. Natural multimodal fusion-based human–robot interaction: Application with voice and deictic posture via large language model. *IEEE Robotics & Automation Magazine*, 2025.
- [6] Chenyang Wang, Jonathan Diller, and Qi Han. Llm for generating simulation inputs to evaluate path planning algorithms. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 176–181. IEEE, 2024. [1](#)
- [7] Yuhang Liu, Yutong Wang, Yuhang Li, Chaoyue Dai, and Fei-Yue Wang. Sensingagent: Advancing vehicular sensing systems for spatiotemporal cognitive intelligence. *IEEE Transactions on Intelligent Vehicles*, 2024. [1](#)
- [8] Tingting Yang, Ping Feng, Qixin Guo, Jindi Zhang, Jiahong Ning, Xinghan Wang, and Zhongyang Mao. Autohama-lm: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models. *IEEE Transactions on Cognitive Communications and Networking*, 2025.
- [9] Zhendong Zhao, Xiaotian Yue, Jiejin Xie, Chuanhong Fang, Zhenzhou Shao, and Shijie Guo. A dual-agent collaboration framework based on llms for nursing robots to perform bi-manual coordination tasks. *IEEE Robotics and Automation Letters*, 2025. [1](#)
- [10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. [1](#)
- [11] Yutaka Matsuo, Yann LeCun, Maneesh Sahani, Doina Precup, David Silver, Masashi Sugiyama, Eiji Uchibe, and Jun Morimoto. Deep learning, reinforcement learning, and world models. *Neural Networks*, 152:267–275, 2022. [1](#)
- [12] Huihui Sun, Hui Jiang, Long Zhang, Changlin Wu, and Sen Qian. Multi-robot hierarchical safe reinforcement learning using autonomous decision-making strategy based on uniformaly ultimate boundedness constraints. *Scientific Reports*, 15(1):5990, 2025. [1](#)
- [13] Lidong Yang, Jialin Jiang, Xiaojie Gao, Qinglong Wang, Qi Dou, and Li Zhang. Autonomous environment-adaptive microrobot swarm navigation enabled by deep learning-based real-time distribution planning. *Nature Machine Intelligence*, 4(5):480–493, 2022. [1](#)
- [14] Huashan Liu, Xiangjian Li, Menghua Dong, Yuqing Gu, and Bo Shen. Robotic motion planning based on deep reinforcement learning and artificial neural networks. *IEEE Transactions on Automation Science and Engineering*, 2024. [1](#)
- [15] Junhong Chen, Zeyu Wang, Ruiqi Zhu, Ruiyang Zhang, Weibang Bai, and Benny Lo. Path generation with reinforcement learning for surgical robot control. In *2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 1–4. IEEE, 2022. [1](#), [2](#)
- [16] Alireza Kheirandish, Duo Xu, and Faramarz Fekri. Llm-augmented symbolic rl with landmark-based task decomposition. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025. [1](#)
- [17] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023.
- [18] Zeyu Wang, Frank P-W Lo, Yunran Huang, Junhong Chen, James Calo, Wei Chen, and Benny Lo. Tactile perception: a biomimetic whisker-based method for clinical gastrointestinal diseases screening. *npj Robotics*, 1(1):3, 2023. [1](#)
- [19] Hsu-Shen Liu, So Kuroki, Tadashi Kozuno, Wei-Fang Sun, and Chun-Yi Lee. Language-guided pattern formation for swarm robotics with multi-agent reinforcement learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8998–9005. IEEE, 2024. [1](#)
- [20] Jiabin Lou, Rongye Shi, Yuxin Lin, Qunbo Wang, and Wenzjun Wu. Talker: A task-activated language model based knowledge-extension reasoning system. *IEEE Robotics and Automation Letters*, 2024. [1](#)
- [21] Zeyu Wang, Frank P-W Lo, Junhong Chen, James Calo, Benny PL Lo, Alex J Thompson, and Eric M Yeatman. An ai-driven bionic whisker system assisting for clinical gastrointestinal disease screening. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2024. [2](#)
- [22] Beili Dong, Junhong Chen, Zeyu Wang, Kaizhong Deng, Yiping Li, Benny Lo, and George Mylonas. An intelligent robotic endoscope control system based on fusing natural language processing and vision models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8180–8186, 2024.
- [23] Ruchi Bagwe, Rashika Natharani, Kiran George, and Anand Panangadan. Natural language controlled real-time object recognition framework for household robot. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1215–1220, 2021.

- [24] Golam Dastagir, Khaled Mushahed Hossain, Md Mahfujul Haque, and Raiyan Bin Gaffar. *NLP-based AI robot for intelligent interaction*. PhD thesis, Brac University, 2024. [2](#)
- [25] Chen Li, Xiaochun Zhang, Dimitrios Chrysostomou, and Hongji Yang. Tod4ir: A humanised task-oriented dialogue system for industrial robots. *IEEE Access*, 10:91631–91649, 2022. [2](#)
- [26] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Gpt-4v(ision) for robotics: Multimodal task planning from human demonstration. *IEEE Robotics and Automation Letters*, 9(11):10567–10574, 2024. [2](#)
- [27] Shyam Sundar Kannan, Vishnunandan L. N. Venkatesh, and Byung-Cheol 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)*, pages 12140–12147, 2024. [2](#)
- [28] Ammar Abdul Ameer Rasheed, Mohammed Najm Abdullah, and Ahmed Sabah Al-Araji. A review of multi-agent mobile robot systems applications. *International Journal of Electrical and Computer Engineering*, 12(4):3517–3529, 2022. [2](#)
- [29] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, et al. Palm-e: An embodied multimodal language model. 2023. [2](#)
- [30] Jair A Bottega, Takashi Tsubouchi, Xinyue Ruan, and Akaihisa Ohya. Are we close to realizing self-programming robots that overcome the unexpected? In *2025 IEEE/SICE International Symposium on System Integration (SII)*, pages 368–374. IEEE, 2025. [2](#)
- [31] Mingming Peng, Zhendong Chen, Jie Yang, Jin Huang, Zhengqi Shi, Qihao Liu, Xinyu Li, and Liang Gao. Automatic milp model construction for multi-robot task allocation and scheduling based on large language models. *arXiv preprint arXiv:2503.13813*, 2025. [2](#)
- [32] Ishika Singh, David Traum, and Jesse Thomason. Twostep: Multi-agent task planning using classical planners and large language models. *arXiv preprint arXiv:2403.17246*, 2024. [2](#)
- [33] Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6695–6702, 2024.
- [34] Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei 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)*, pages 2081–2088, 2024.
- [35] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299. IEEE, 2024. [2](#)
- [36] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4311–4317. IEEE, 2024.
- [37] Callie Y Kim, Christine P Lee, and Bilge Mutlu. Understanding large-language model (llm)-powered human-robot interaction. In *Proceedings of the 2024 ACM/IEEE international conference on human-robot interaction*, pages 371–380, 2024. [2](#)
- [38] Peihan Li, Zijian An, Shams Abrar, and Lifeng Zhou. Large language models for multi-robot systems: A survey. *arXiv preprint arXiv:2502.03814*, 2025. [2](#)
- [39] Wenyu Zhang, Jingyao Gai, Zhigang Zhang, Lie Tang, Qingxi Liao, and Youchun Ding. Double-dqn based path smoothing and tracking control method for robotic vehicle navigation. *Computers and Electronics in Agriculture*, 166:104985, 2019. [2](#)
- [40] Bin Wu and C Steve Suh. Deep reinforcement learning for decentralized multi-robot control: A dqn approach to robustness and information integration. In *ASME International Mechanical Engineering Congress and Exposition*, volume 88636, page V005T07A035. American Society of Mechanical Engineers, 2024. [2](#)
- [41] Yoko Sasaki, Syusuke Matsuo, Asako Kanezaki, and Hiroshi Takemura. A3c based motion learning for an autonomous mobile robot in crowds. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 1036–1042, 2019. [2](#)
- [42] Hamid Taheri, Seyed Rasoul Hosseini, and Mohammad Ali Nekoui. Deep reinforcement learning with enhanced ppo for safe mobile robot navigation. *arXiv preprint arXiv:2405.16266*, 2024. [2](#)
- [43] Weixin Quan, Wenbo Zhu, Qinghua Lu, Lufeng Luo, Kai Wang, and Meng Liu. Obstacle avoidance control method for robotic assembly process based on lagrange ppo. In *International Conference on Cognitive Systems and Signal Processing*, pages 16–26. Springer, 2023.
- [44] Shengjia Shao, Jason Tsai, Michal Mysior, Wayne Luk, Thomas Chau, Alexander Warren, and Ben Jeppesen. Towards hardware accelerated reinforcement learning for application-specific robotic control. In *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 1–8, 2018. [2](#)
- [45] Phuc Dang Thi, Chinh Nguyen Truong, and Hieu Dau Sy. Rac-sac: An improved actor-critic algorithm for continuous multi-task manipulation on robot arm control. In *Proceedings of the 12th International Symposium on Information and Communication Technology*, pages 824–830, 2023. [2](#)
- [46] Yuan Zhuang, Yi Shen, Zhili Zhang, Yuxiao Chen, and Fei Miao. Yolo-marl: You only llm once for multi-agent reinforcement learning. *arXiv preprint arXiv:2410.03997*, 2024. [2](#)
- [47] Arne Wahrburg and Kim Listmann. Mpc-based admittance control for robotic manipulators. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7548–7554, 2016. [2](#)
- [48] Gian Paolo Incremona, Antonella Ferrara, and Lalo Magni. Mpc for robot manipulators with integral sliding modes generation. *IEEE/ASME Transactions on Mechatronics*, 22(3):1299–1307, 2017. [2](#)
- [49] RB Shyam, Zhou Hao, Umberto Montanaro, and Gerhard Neumann. Imitation learning for autonomous tra

- jectory learning of robot arms in space. *arXiv preprint arXiv:2008.04007*, 2020. [2](#)
- [50] José Pedro Carvalho and A. Pedro Aguiar. Deep reinforcement learning for zero-shot coverage path planning with mobile robots. *IEEE/CAA Journal of Automatica Sinica*, pages 1–16, 2025. [2](#)
- [51] Ziqi Yang, Ruiyang Zhang, Junhong Chen, Xuhui Zhou, Yunxiao Ren, Ziyue Tong, and Benny Lo. Automated trajectory generation for robotic surgical tasks. In *2024 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 39–44. IEEE, 2024. [2](#)
- [52] Ruiyang Zhang, Junhong Chen, Zeyu Wang, Ziqi Yang, Yunxiao Ren, Peilun Shi, James Calo, Kyle Lam, Sanjay Purkayastha, and Benny Lo. A step towards conditional autonomy-robotic appendectomy. *IEEE Robotics and Automation Letters*, 8(5):2429–2436, 2023. [2](#)