Biomimetic Intelligence and Robotics 6 (2026) 100334

Contents lists available at ScienceDirect

Biomimetic Intelligence and Robotics

journal homepage: www.keaipublishing.com/en/journals/biomimetic-intelligence-and-robotics/

Research Article

D-RMGPT: Robot-assisted collaborative tasks driven by large multimodal models

Matteo Forlini [a], Mihail Babcinschi [b], Giacomo Palmieri [a], Pedro Neto [b,*]

[a] Department of Industrial Engineering and Mathematical Sciences, Polytechnic University of Marche, Ancona 60131, Italy
[b] Centre for Mechanical Engineering, Materials and Processes (CEMMPRE), ARISE, University of Coimbra, Coimbra 3030-788, Portugal

ARTICLE INFO

Article history:
Received 21 October 2025
Revised 16 March 2026
Accepted 26 March 2026
Available online 17 April 2026

Keywords:
Collaborative robotics
Large multimodal models
Human–robot interaction

ABSTRACT

Collaborative robots are increasingly popular for assisting humans at work and daily tasks. However, designing and setting up interfaces for human–robot collaboration is challenging, requiring the integration of multiple components, from perception and robot task control to the hardware itself. Frequently, this leads to highly customized solutions that rely on large amounts of costly training data, diverging from the ideal of flexible and general interfaces that empower robots to perceive and adapt to unstructured environments where they can naturally collaborate with humans. To overcome these challenges, this paper presents the Detection-Robot Management GPT (D-RMGPT), a robot-assisted planner based on Large Multimodal Models (LMM) to assist inexperienced operators in tasks without requiring any markers or previous training. D-RMGPT is composed of DetGPT-V and R-ManGPT. DetGPT-V, based on GPT-4V(vision),now replaced by GPT-4o, perceives the surrounding environment through one-shot analysis of prompted images from the current scenario, identifying the surrounding objects, human actions and interpreting speech commands. R-ManGPT, based on GPT-4 and aided by a 3D CNN-LSTM network, acts as operations and robot task manager, planning the next actions, and generating discrete robot actions. Planning precedence dependencies between tasks is solved by giving the system a single image with the full list of precedence relationships. Experimental tests on assembling a toy aircraft demonstrated that D-RMGPT is flexible and intuitive to use, achieving an assembly success rate of 83% while reducing the assembly time for inexperienced operators by 33% compared to the manual process. The cooking and food preparation assistance scenario demonstrated that D-RMGPT successfully anticipated operator needs and assisted in producing a pizza from selected ingredients, achieving a success rate of 87%. http://robotics-and-ai.github.io/LMMmodels/.

1. Introduction

Collaborative robots are increasingly present and widespread in a wide variety of application domains, aiming to reduce human effort, structure collaborative work, and enhance productivity. The effectiveness of these robots is highly dependent on the intuitiveness and robustness of the human–robot interfaces and collaborative processes. Significant advancements have been achieved in the field of human–robot interaction (HRI) in recent years. However, these advancements often result in structured applications tailored to specific tasks that are challenging to configure and lack flexibility. Moreover, key elements of a HRI system, such as perception and reasoning, often rely on deep learning or reinforcement learning methodologies that require large amounts of training data. In this context, advances in HRI require general, reliable and integrated solutions to empower robots to perceive the surrounding environment, learning from human co-workers and previous experiences, and reasoning in uncertain conditions.

Recent advancements in foundation models such as Large Language Models (LLM), Vision Language Models (VLM) and Large Multimodal Models (LMM), have demonstrated significant potential to support robot planning activities and related HRI applications [1]. While LMM trained in specific application datasets offer superior performance, they can limit the general applicability and flexibility of the robotic system [2]. On the other hand, using off-the-shelf models can strike a good balance between performance and general applicability, without the need for additional training data and model retraining [3,4].

Task assistance, flexible task planning, and decision-making are essential factors for empowering collaborative robots and enabling their successful widespread adoption [5]. Assembly is one of the most common tasks in manufacturing, spanning from fully automated systems to semi-automated to fully manual assembly. Like in assembly manufacturing, cooking and food preparation assistance is a highly desired functionalitiy for robots, both in

* Corresponding author.
E-mail address: pedro.neto@dem.uc.pt (P. Neto).

https://doi.org/10.1016/j.birob.2026.100334

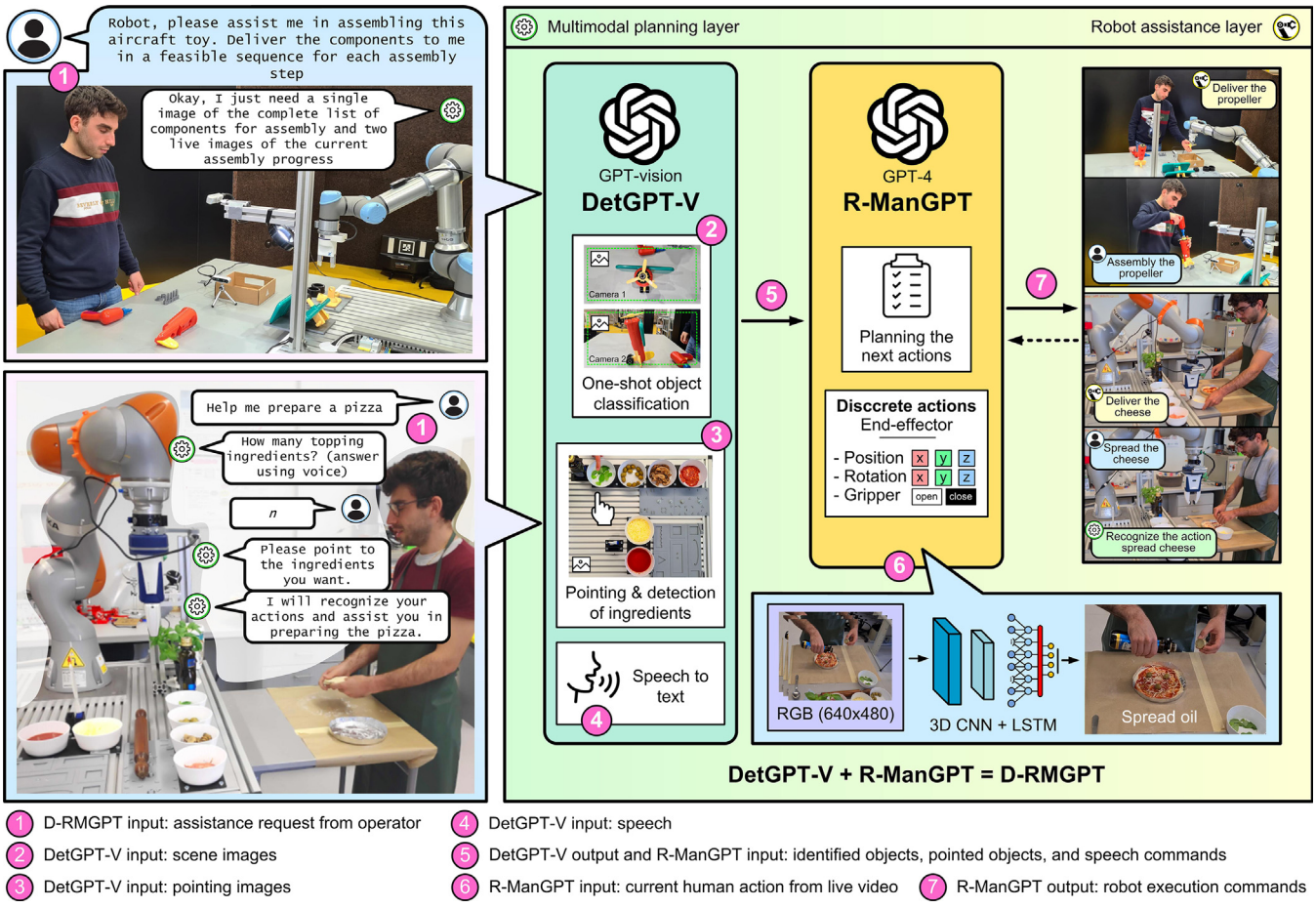

Fig. 1. The D-RMGPT architecture comprises the detection module DetGPT-V and the robot management and planner module R-ManGPT. These modules, based on GPT-4V(ision) and GPT-4, enable an inexperienced operator to successfully complete a task assisted by a collaborative robot.

the food industry and in domestic settings [6]. The automation of these processes can be difficult to achieve, as many tasks require dexterity skills. This scenario opens the door to human–robot collaborative work, leveraging the best abilities of both robots and humans for a more productive and less error-prone workflow [7]. In addition, each actor can focus on tasks where they perform best, humans can handle tasks requiring high dexterity, while robots can perform monotonous tasks demanding physical effort. Having robots doing such tasks is key in the current context of skilled labor shortages. However, the integration of robots in collaborative assembly operations is challenging due to the high variability of the process, the required dexterity, and the need for reliable perception systems. These are limiting factors for the deployment of collaborative robots. Multimodal interfaces should enable robots to understand their surroundings, recognize human actions and intentions, and interpret speech commands. Adaptive planning should support robot autonomy by determining the next action steps based on the analysis of the current scenarios, reasoning within the actual context, and anticipating potential occurrences without relying on extensive data collection.

This study proposes the Detection-Robot Management GPT (D-RMGPT), an adaptable robot-assisted planner based on embedded knowledge of off-the-shelf GPT-4 and GPT-4V(ision). This system is designed to assist inexperienced operators in human–robot collaborative tasks without requiring any markers or prior local training. D-RMGPT comprises two main components: the detection module DetGPT-V and the robot management and planner module R-ManGPT, Fig. 1. DetGPT-V perceives the environment through one-shot analysis of prompted images from the current scenario, identifying the surrounding objects, human actions (e.g., pointing and tasks) and interpreting human speech commands. For instance, in an assembly process, it requires only a single image with the full list of components and their assembly precedence dependencies as input, eliminating the need to identify missing components. In a cooking preparation scenario, it recognizes ingredients being pointed to by the human and speech commands. R-ManGPT acts as operations and robot task manager, planning the next actions, such as identifying feasible components for assembly or selecting ingredients to deliver, and generating discrete robot actions. A 3D CNN-LSTM network is used to classify human actions and relay them to R-ManGPT. This work resulted in the following contributions:

- D-RMGPT, a flexible robot-assisted planner based on GPT-4 and GPT-4V(ision) to assist inexperienced operators in human–robot collaborative tasks, guiding the operators while adapting to their choices. It can adapt to complete a task even when the operator deviates from the suggested actions, which makes the online planning process more challenging.
- A detection module, DetGPT-V, perceives the surrounding environment through one-shot analysis of prompted images and interpret speech comands. DetGPT-V's component detection abilities compare favorably with commonly used VLM-based object detectors such as ViLD and OWL-ViT.
- A robot operation and task manager module, R-ManGPT, dynamically plans the next robot's discrete actions to execute

and generates the respective robot's commands. It is easy to finetune for new application domains, being camera and robot independent.
- D-RMGPT significantly improves the intuitiveness, flexibility, availability and general applicability of human–robot collaborative work while reducing the total work time for inexperienced operators.
- D-RMGPT offers a unified multimodal approach to collaborative robotics, integrating perception and reasoning within a single off-the-shelf LMM (GPT-4V).
- Unlike prior systems that depend on external detectors, RL-trained skills, or large fine-tuning datasets, D-RMGPT performs zero-shot component detection, continuous image-grounded replanning, and seamless adaptation to operator actions.

1.1. Related work

In recent years, foundation models have seen widespread usage in different applications, ranging from text recognition, text generation, coding, and even extending to the new frontier of image and video classification and generation [8]. Due to their intrinsic characteristics of perception, reasoning and semantic knowledge [9,10], foundation models such as LLMs [11] and VLMs [12] are promising intelligent agents for robotics [1,13] and manufacturing [14]. They enable robots to understand and reason about the surrounding environment, plan activities, support decision-making, reasoning and planning [15]. Human–robot interfaces can be easily handled via a simple prompt request, with the potential of making the interactive process more intuitive, and adaptable to the needs of different users and application scenarios. However, they also present challenges concerning the understanding of real-world environments, dealing with constraints, and perceiving the physical state of a robot. Often, these systems lack important implicit information, leading to inaccurate and insecure actions that result in task failure. Recent studies address these challenges by providing extra awareness tools to the system, such as detectors like Open Vocabulary Object Detection or value functions used as grounding for LLMs [16–19]. The release of GPT-4 and GPT-4V(ision) makes it possible to combine perception and reasoning for both language and images, creating a synergy that achieves good performance when applied in HRI [20]. LLM can facilitate human–robot interfacing and learning by integrating speech commands through natural language for assembly task management [21], augmenting data samples for few-shot robot learning [22] and robot locomotion control [23]. LLM-based planners can be personalized with human preferences through iterative planning, recurring to an optimization pipeline that integrates imitation learning and reinforced Self-Training [24]. An LLM-powered human–robot collaborative agent enables flexible collaboration, supported by long term memory and continuous instruction understanding, and demonstrated in complex aerospace assembly tasks [25]. A recent study proposes an LLM-based agent framework aimed at enhancing proactive human–robot collaboration in smart manufacturing environments. Results indicate improved adaptability in dynamic environments [26].

In computer vision, deep learning methods have demonstrated state-of-the-art performance in object detection and classification [27–29]. However, these methods present a number of challenges related to their reliability and require large and specific datasets for training [30]. Foundation models such as VLM, trained on massive amounts of general text-image pairs, can help mitigate these limitations by providing text descriptions of images [31,32]. Some models use contrastive learning by computing the similarity between text and image embeddings using textual and visual encoders [33,34]. CLIP-Fields learns mapping from spatial locations, raw RGB-D and odometry data, to semantic embedding vectors [35]. SimVLM is a VLM architecture that uses a transformer encoder to learn image-prefix pairs and a transformer decoder to generate an output text-based sequence [36]. VLM can identify and detect different components in an image by processing a text prompt that indicates which components need to be investigated in the scene [37]. Robots can use this information to grasp objects in a scene they are operating in for the first time without specific training. Grounding Languages-Image-Pre-Training (GLIP) unifies object detection and phrase grounding, which involves identifying the correspondence between phrases in a sentence and objects in an image by passing both a text prompt and images to the system [38,39]. Open-World Language Vision Transformer (OWL-ViT) is a vision-language model designed to handle open-vocabulary object detection using transformers [12]. MDETR is a framework for object detection that uses a convolutional backbone to extract visual features and the language model RoBERTa to extract text features [40].

The characteristics of foundation models make them a key element in the advancement of robotics [13,41]. The SocraticModel integrates LLMs and VLMs to combine their different commonsense knowledge, exchange information with each other, and capture new multimodal capabilities without requiring finetuning [42]. This model has been applied in robotic planning and decision-making, using ViLD for object detection in the scene and LLM for task planning [37,43]. SayCan combines low-level skills with LLM, enabling the LLM to provide detailed instructions on how to perform an abstract high-level task in a feasible way for a robot [44]. LLM ability to reason over natural language, without any additional training, allows them to guide robots in performing manipulation tasks [45], to act as zero-shot human models for HRI prediction of human behaviors [46], or to decompose high-level commands into sequences of robot motions using a prompted GPT-4 [47,48]. Prompting pre-trained VLM facilitates reasoning in zero-shot and few-shot manners to solve robotic manipulation [49]. Mobile robot motion planning has been addressed using LLM [50]. Knowing the table arrangement objects and their dimensions, GPT-3 can provide the relative position of each item on the table to the robot motion planner. Recently, robot navigation has been achieved by combining environment understanding and common-sense reasoning of long-context VLMs with a low-level navigation policy [51,52]. LLM text-davinci-003 generates robot action commands to interactively perceive an environment from multimodal sensor feedback, selecting the object to grasp [53]. An interesting study shows that if pre-trained LLM are large enough and prompted appropriately, they can effectively decompose high-level tasks expressed in natural language into mid-level plans without any further training [43]. Primitive trajectories for robot manipulation can be synthesized by using LLMs to infer affordances and constraints [54]. Leveraging LLM code-writing capabilities, they can interact with VLM to compose 3D value maps. PaLM-E integrates the LLM PaLM with a vision transformer to transfer knowledge from visual-language domains into embodied reasoning [11,55]. This enables the model to reason about text and other modalities jointly. Using GPT-4V(ision), with an image of the current scene and a list of possible sub-tasks as input, a sequence of robot tasks can be planned to reach a given goal [56]. RoboGPT is a framework in which GPT-3.5 is used to generate demonstrations that are then used to train a robot [57].

Recently, the functionalities and applicability of LLM have grown exponentially. However, implementing LLM-based robotic systems faces significant challenges. Robots are often limited to performing simple tasks and are highly sensitive and dependent to prompt design. They also exhibit reduced performance in tasks requiring numerical, physical, or spatial reasoning, and their integration and coordination with other perception tools remain complex.

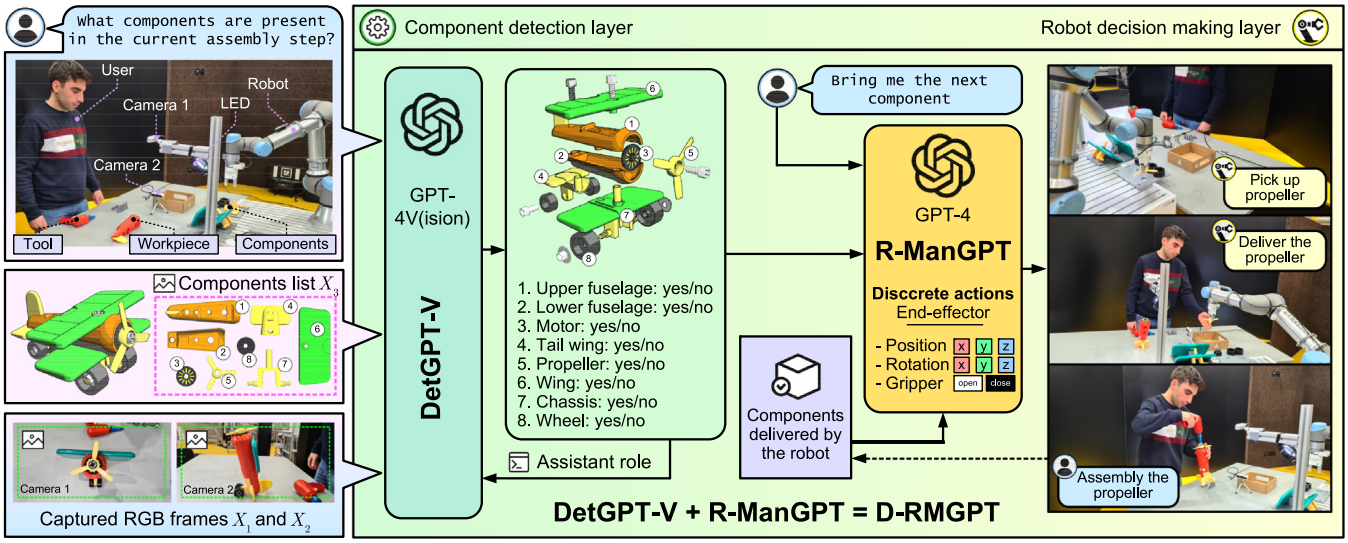

Fig. 2. D-RMGPT architecture for the toy aircraft assembly application with the detailed inputs and outputs for each submodule, DetGPT-V and R-ManGPT. The inputs to the system are data acquired from two cameras (Camera 1 and Camera 2) and the components list.

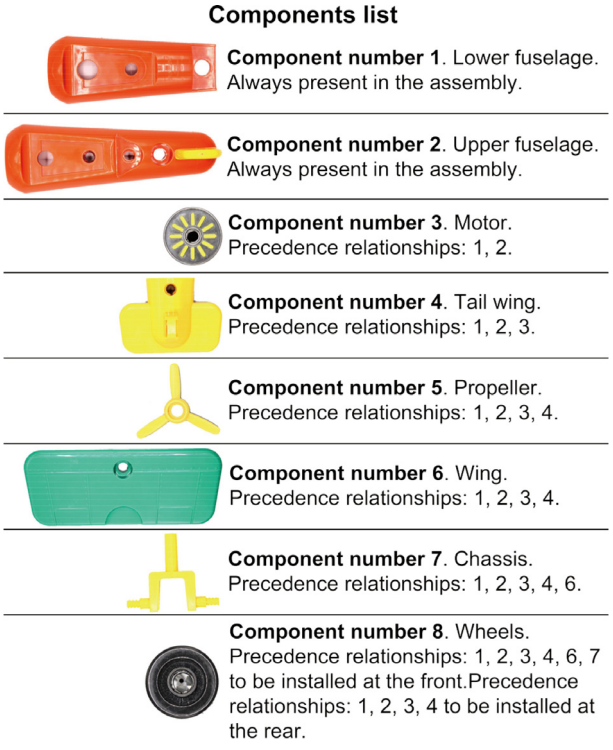

Fig. 3. Component list image X_3. It includes the components picture, number, description and assembly precedence dependencies, i.e., the components that need to be assembled before the actual component.

2. Detection-robot management GPT (D-RMGPT)

The development of D-RMGPT was driven by the need to assist inexperienced operators in human–robot collaborative tasks. Task planning for these applications, such as assembly and cooking applications, can be challenging due to precedence dependencies and the flexibility required to accommodate human preferences. Although D-RMGPT shares a common architecture and is of general applicability, it is designed to adapt to the specific requirements of each application.

2.1. Assembly process

A toy aircraft from the Yale-CMU-Berkeley object and model dataset [58] is used as assembly workpiece to assess the performance of the proposed robot-assisted assembly planner. The aircraft is composed of eight different components, plus a power tool, screws, and nuts available to the operator, Fig. 2. The complete assembly can be achieved by following a number of different assembly sequences. However, if certain assembly precedence dependencies are not respected, it is not possible to successfully complete the assembly of the aircraft. GPT-4 has demonstrated strong performance in solving relational planning problems in scenarios where precedence relationships are either non-existent or easily inferred through common-sense reasoning [59,60]. Some planning problems require managing precedence dependencies between tasks. LLMs struggle to address such problems without additional information to ensure task execution efficiency. Recent studies have begun tackling this issue by integrating techniques such as linear programming and dependency graphs [61]. Here, we address the problem by giving the system a single image with the full list of components and their assembly precedence relationships as input. The system detects installed components by comparing the current assembly scene with the component list. Reasoning based on precedence relationships helps validate the presence of components and reduces the likelihood of hallucinations. The model is not aware of the assembly process and the functionality of each component, as no assembly instructions are provided.

The assembly process is recommended to start with the two fuselage components, component #1 and component #2, Fig. 3. The remaining components are placed in a storage area from which the robot can pick up and deliver them close to the operator, Fig. 2. The wheels are divided into a rear pair and a front pair. Since they are identical in color and shape, it is expected that the detection module DetGPT-V will find it challenging to distinguish between them and detect if the wheels are placed at the front or rear of the toy aircraft.

The operator can follow the assembly order defined by the D-RMGPT, or at any step of the assembly, the operator can decide to

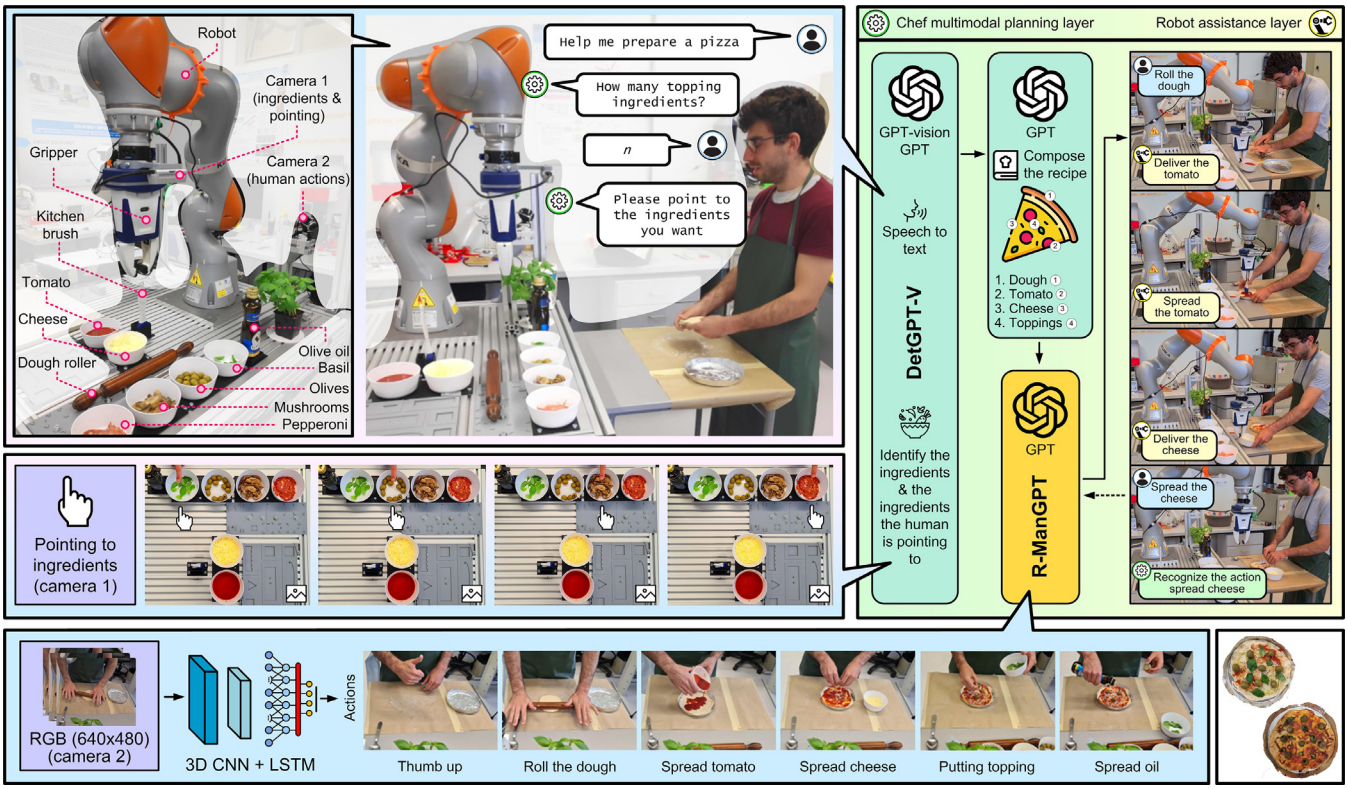

Fig. 4. D-RMGPT architecture for the cooking pizza application with the detailed inputs and outputs for each submodule (DetGPT-V and R-ManGPT). Environmental data are captured by two cameras to detect ingredients, pointing gestures, and human actions, as well as by a microphone to capture human voice.

not follow the D-RMGPT recommendation and choose a different component to assemble on their own initiative. The D-RMGPT should be able to adapt to this unexpected scenario and continue the planning-delivering process to reach the complete assembly.

2.2. Cooking pizza

Cooking pizza involves multiple steps that can be easily customized based on the human's choices. The variability in pizza making, with its numerous possible ingredient combinations, demands a highly flexible planner. D-RMGPT begins each pizza preparation with the human's decision on how to customize the recipe. The human selects, via speech command, the number of toppings to include on the pizza, Fig. 4. The choice of toppings ranges from zero to four, though the framework can accommodate a larger selection if needed. Once the number of toppings is set, the human selects specific ingredients by pointing at them. DetGPT-V analyses snapshots of the human's hand to recognize which ingredient(s) they are pointing to and generates a recipe accordingly. Operator gestures were employed for ingredient selection to validate the framework's robustness in multimodal input perception and to ensure a more intuitive user interaction. Based on the recipe and the online classification of the human's cooking actions (Video in supplementary materials), the task planner, R-ManGPT, identifies the next action to perform and commands the robot to deliver the required kitchen utensils or ingredients from the storage area, Fig. 4. This anticipatory approach enables the robot to suggest and prepare the subsequent steps proactively. From a practical perspective, once the recipe is defined, the preparation sequence begins when the operator signals readiness with a thumbs-up gesture. Having a ball of pizza dough in the working table, the operator rolls it out using a rolling pin delivered by the robot. Next, the operator adds the ingredients provided by the robot: tomato sauce, mozzarella, selected toppings, and olive oil. Additionally, the robot assists the operator in spreading the tomato sauce evenly over the dough.

2.3. Proposed approach

D-RMGPT was conceived to be an adaptable robot-assisted planner. DetGPT-V and R-ManGPT are LMM-based frameworks that utilize the embedded knowledge of off-the-shelf GPT-4V(ision) and GPT-4 to reason and generate robot actions, enabling human assistance. DetGPT-V ensures continuous perception and situational awareness of the task and human state, while R-ManGPT operates at a deliberative planning level, translating this information into discrete, high-level robot actions. This separation of perception and planning decouples multimodal inference from robot execution, contributing to both system stability and practical usability in real-world collaborative settings.

2.3.1. DetGPT-V

The detection module DetGPT-V, based on off-the-shelf GPT-4V, utilizes one-shot images captured at each working step, from a single or multiple cameras, to assess the current status of the scene under analysis. For example, it can determine which components are already assembled, identify the ingredients the human is pointing to, or recognize which ingredients are still available in the storage area. No textual descriptions of the elements to detect or process sequencing are provided. In applications with assembly precedence, as in the toy aircraft assembly, only a single image X_3 listing all the components and their assembly precedence dependencies is required as input at the beginning of the process to fine tune GPT-4V, Fig. 3. For this application, two cameras are positioned to capture top and side views of the workbench where the aircraft is assembled, X_1 and X_2, Fig.

2. The aircraft workpiece must be placed on the workbench in a position where all components are visible from at least one camera, minimizing occlusions. DetGPT-V generates a list of components present in the current assembly stage, which is saved to a text file and passed to the next iteration through the assistant role. This list is then provided to R-ManGPT. In the cooking pizza application, DetGPT-V collects the number of ingredients and uses camera images to detect the ingredients selected by the human through pointing, Fig. 4. DetGPT-V continuously monitors which ingredients and kitchen utensils remain available in the storage area, as the human operator can independently pick and use a tool or ingredient before it is suggested by the system. This feature is particularly advantageous for experienced operators who may choose to deviate from the suggested cooking sequence.

2.3.2. R-ManGPT

R-ManGPT, based on off-the-shelf GPT-4, plans the next feasible actions and generated the corresponding robot actions. These actions may include identifying feasible components for assembly, selecting ingredients to deliver, or performing specific tasks such as using a pastry brush to spread tomato sauce, Figs. 2 and 4. Robot actions are defined by the robot end-effector poses (position and orientation of the objects are fixed and known) and the gripper state (open or close) for pick-and-place operations. R-ManGPT maintains a memory of actions already performed by the robot. A 3D CNN-LSTM network classifies human actions and, if required, relay them to R-ManGPT, depending on the application. The robot operates not only in an assistance role but also in an anticipatory role, enabling a smoother workflow by predicting the operator's needs. This approach enhances productivity and reduces human effort, ensuring a more fluid and collaborative task execution.

Algorithm 1 outlines the implementation of the D-RMGPT architecture for the assembly application. For the pizza cooking application, the structure is similar, with an additional step where the operator sets the process variables.

Algorithm 1 D-RMGPT algorithm

Require: $X_1, X_2, X_3, \mathcal{L}, det_{t=0}$ ▷ X_m are the input images, $\mathcal{L}$ the prompt request, and $det_{t=0}$ the first components list.
$t = 1$
$avail_{t=0} = \{1, ..., n\}$ ▷ List of all the available components at the initial step.
$brought_{t=0} = \varnothing$ ▷ Components already brought by the robot.
while $avail_{t-1} \neq \varnothing$ **do**
  $det_t = LMM(X_1, X_2, X_3, \mathcal{L}, det_{t-1})$ ▷ DetGPT-V components identification, where det_t is the list of the detected components.
  $bring_t = LMM(det_t, brought, avail_{t=0}, action_{t-1})$ ▷ R-ManGPT plans the next component to be picked up and delivered by the robot.
  $brought_t = bring_t \cup brought_{t-1}$
  $avail_t = avail_{t=0} - (det_t \cup brought_t)$ or $avail_t = det_t$ ▷ It depends on which user-case it is considered.
  $t = t + 1$
end while

2.3.3. Prompt structure

We designate an aircraft component, ingredient, or cooking tool simply as a component. In addition, we designate each working step as an assembly step, whether for the toy aircraft assembly or the pizza cooking process.

The DetGPT-V prompt structure takes at maximum three images, X_1, X_2 and X_3, as input, which are analyzed by GPT-4V to detect the components present. In the case of the toy aircraft assembly, in the user role, the three images are loaded, and prompt questions are posed to determine whether each component is present in the two current state images, X_1 and X_2, requiring a YES or NO response, Fig. 5.

The components list image X_3, Fig. 3, with a size of 768 × 2048 pixels, is analyzed in high detail mode so that GPT-4V does not decrease its resolution by downsampling. It has a cost of 1365 tokens. The current assembly step is captured by the two cameras, which capture top and side images, X_1 and X_2, with a resolution of 680 × 480 pixels, each costing 425 tokens. In contrast, in the pizza cooking scenario, only the image X_1 (Fig. 5, img_camera1) is provided, depicting the current state of the storage area, along with the ingredients and kitchen utensils that remain available.

An example of the response provided by the system is then reported to the assistant role so that the same deterministic output is always given. To get more accurate and faster responses, a question is asked for each component, indicating YES or NO as possible answers. The output is a list where each component is indicated as present (YES) or not (NO). The prompt structure was set up following the instructions and suggestions in [62]. Asking to investigate the presence of each individual component in separate queries within the user role allows the identification problem to be decomposed into several subtasks, one for each component, yielding better results in terms of precision and recall. Fig. 6 illustrates the prompt structure used for image explanations in both the assembly and cooking scenarios.

In the pizza cooking scenario, there is a preliminary stage that allows the operator to configure certain variables for the process setup. The first part of this stage manages the speech input, where the operator specifies the number of toppings. This input is processed using the GPT model *whisper-1* to transcribe the recorded audio file, indicating the number of toppings, which can range from zero to four. The transcription is post-processed via GPT-4 model to handle eventual ambiguities and extract as information only the number. The second part identifies which ingredient the operator is pointing to in a given image. The input is a 640 × 300 pixel image capturing the operator's gesture. This ingredient selection routine is repeated as many times as the number of toppings specified in the audio recording. The output is the name of the selected ingredient, such as mushrooms, basil, and so on. Fig. 7 shows the prompt structure.

The R-ManGPT prompt structure is illustrated in Fig. 8. The user content is asked what is the next action the robot should perform to assist the operator, who is currently operating in a specific phase of the process, denoted as <*phase_name*>. The <*phase_name*> serves as input for the system and can refer to the assembly of a particular component or an action related to the pizza cooking process. In the assistant role, the desired output type is specified, including commands to be sent to the robot. The global process for selecting the next action is divided into distinct reasoning steps, Fig. 9. This approach enables accurate results, even in complex reasoning processes such as those proposed, which while similar, exhibit some differences.

In the cooking pizza process, the ingredients available in the storage area are directly detected, whereas in the aircraft assembly, the components that have already been assembled are identified. Additionally, the pizza cooking process incorporates anticipatory skills enabled by inputs from automatic action recognition. This allows the system to initiate the next action while the operator is completing the previous one. In contrast, in the aircraft assembly process, anticipation is not permitted. The system must wait for the operator to complete the installation of a component before suggesting the next robot action. In practice, the <*phase_name*> is $<$ *using* + *component_name* $>$ or <*continue_to_next_step*> in the case of thumb up action. In the aircraft assembly the <*phase_name*> is $<$ *receiving*+*component_number* $>$. Fig. 9 shows the content provided in the action list file of the R-ManGPT. Regarding token usage, in the aircraft assembly, which

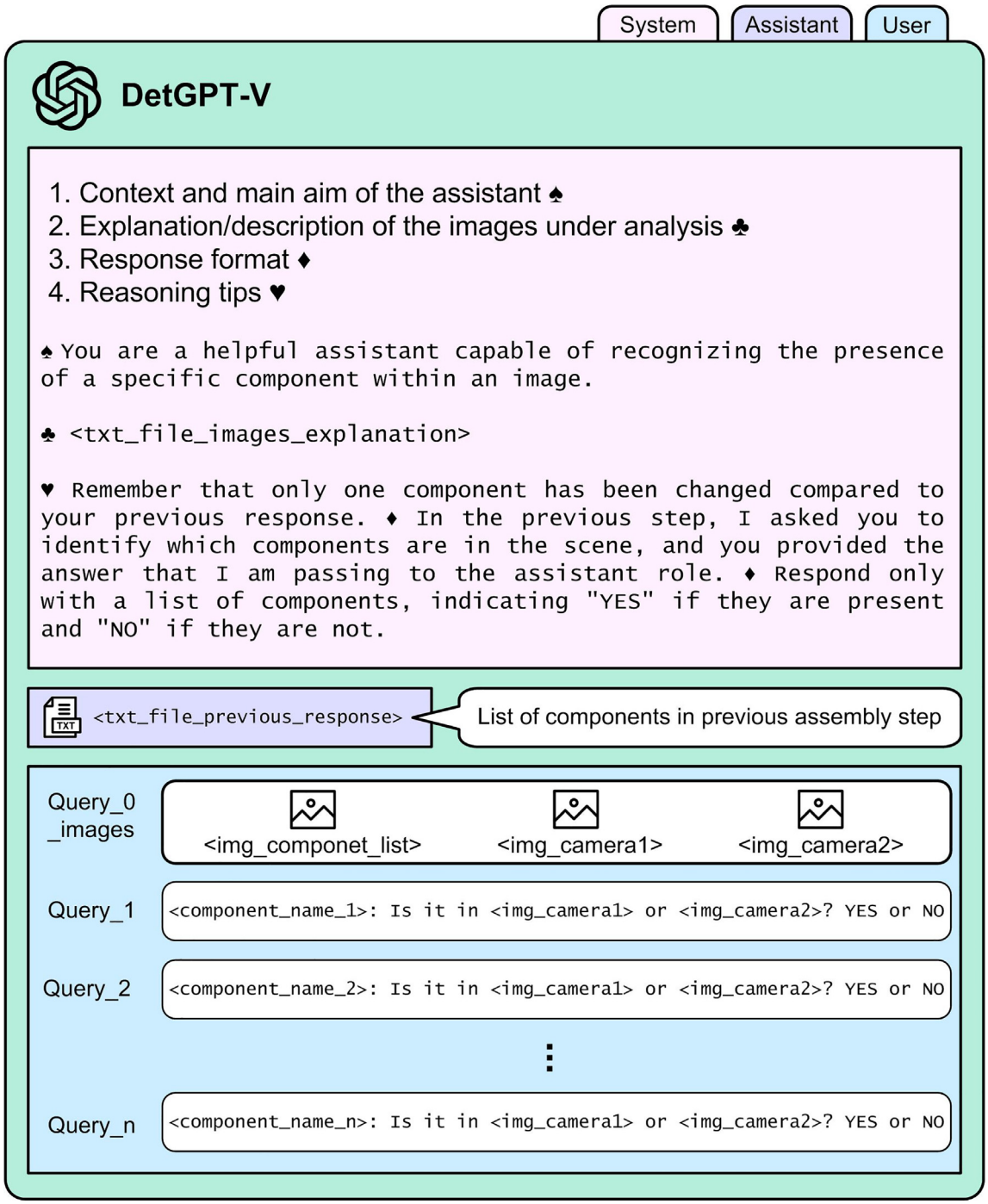

Fig. 5. Prompt structure for the detection module DetGPT-V.

is the most demanding application due to the use of three images in detailed mode, each request to the server consumes about 2215 + 710 input tokens and 120 output tokens. This cost is incurred at every step of the assembly process. Token usage can be reduced by optimizing the number and resolution of input images, even changing the detailing image processing level. The conversion from tokens to cost depends on the specific OpenAI contract. At the time we conducted this work, the cost ranged from approximately 0.0225 to 0.03 USD per server request.

It is worth noting that the basic structure of the prompts used in DetGPT-V and R-ManGPT is the same for both applications (see Figs. 5, 8), since they share the common goal of assisting the operator via a robot during assembly tasks. However, some parts of the prompts must be adapted to the specific scenario at hand (e.g., different components, different robot actions, Figs. 6, 9). This does not imply that a separate pretraining procedure is required for each scenario, as no additional pretraining is needed apart from the action classification module based on a 3D CNN-LSTM. In this work, we present a prompt design methodology and a framework built around a detection stage followed by an action planning stage. Its functionality was tested on two application scenarios, with prompts adapted to the specific requirements of each one.

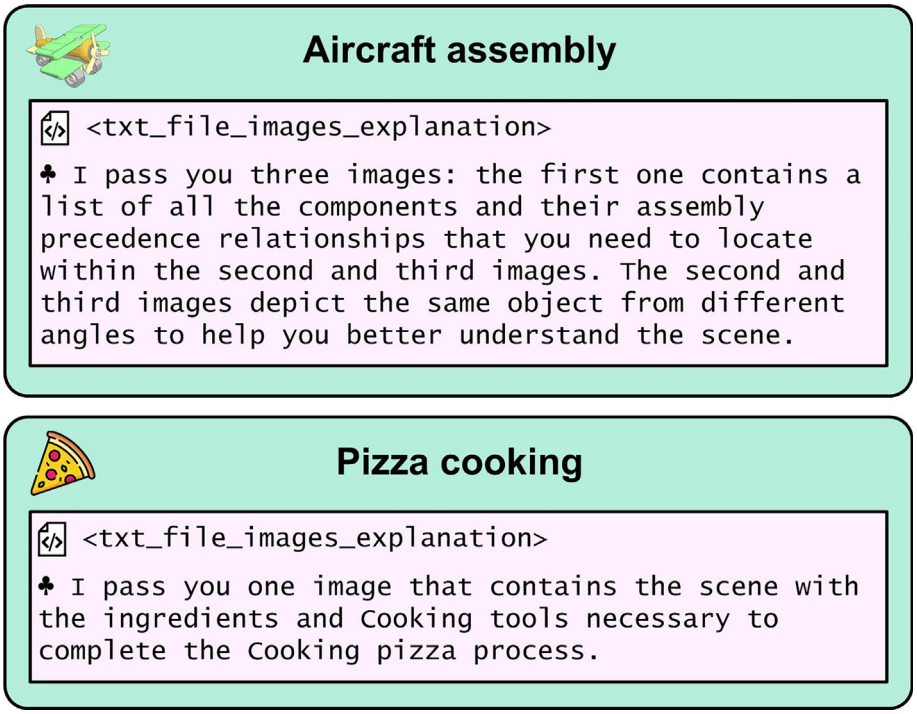

Fig. 6. DetGPT-V prompt structure used for image explanations in both the assembly and cooking scenarios.

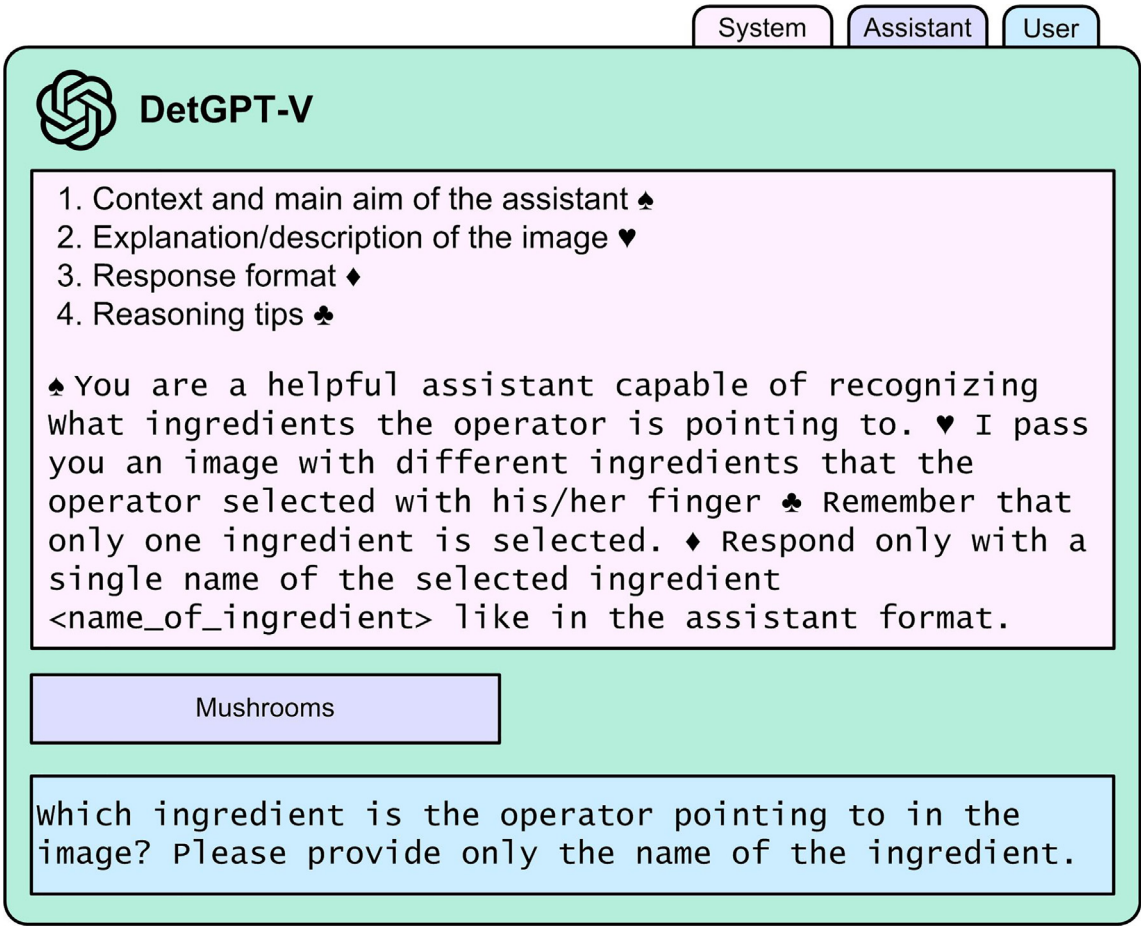

Fig. 7. Prompt structure for detecting selected ingredients.

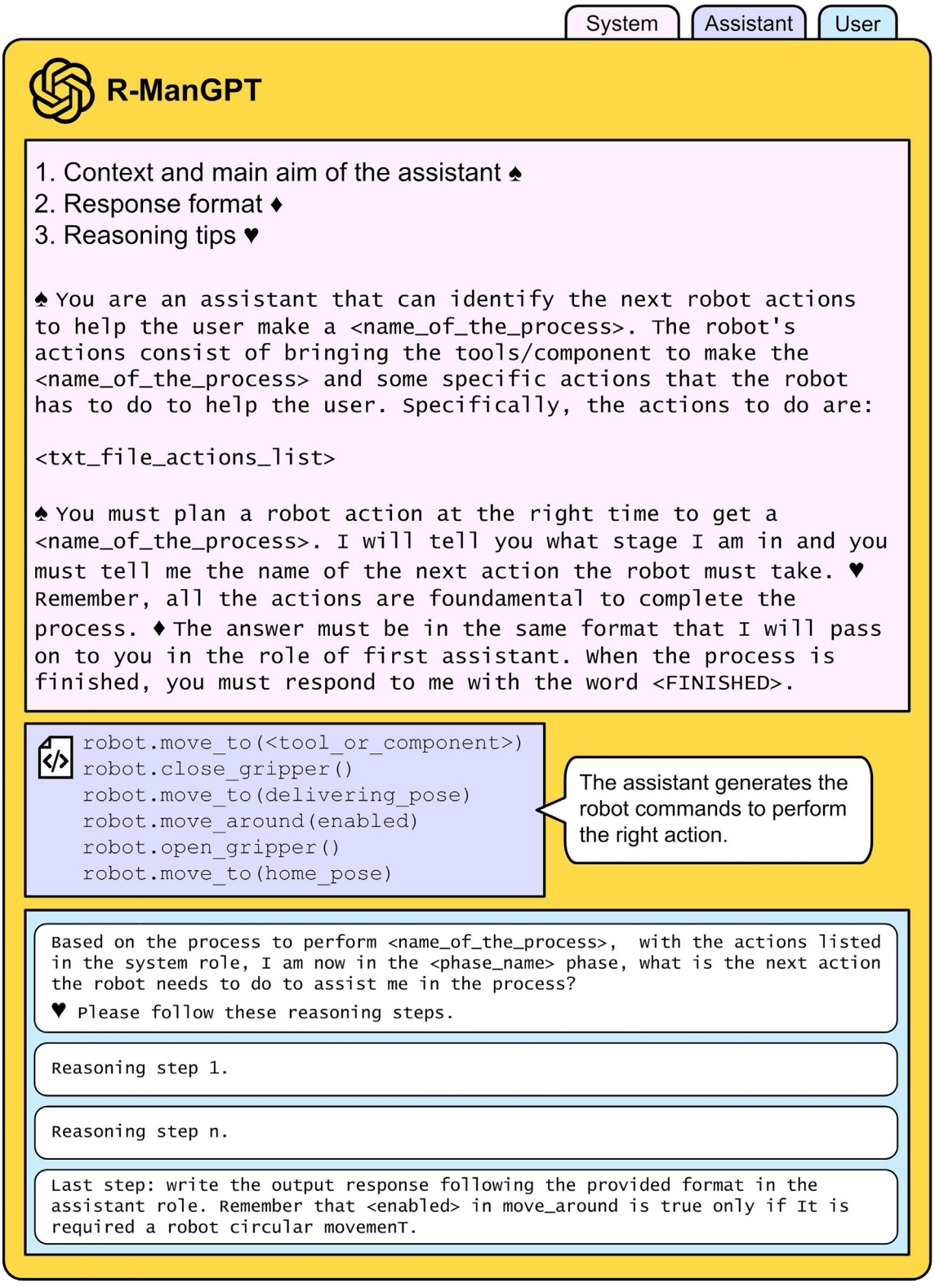

Fig. 8. Prompt structure of the robot management module R-ManGPT.

2.3.4. Action recognition

An3D CNN-LSTM network was implemented due to its reported high accuracy and reliability in recognizing dynamic human gestures in real-time, along with its ability to train a custom recognition model. The 3D CNN extracts spatio-temporal features, an LSTM layer models longer temporal relations, and a SoftMax layer generates output classifications. The model was pre-trained with a dataset of short, densely labeled video clips of human actors performing generic hand gestures in front of a camera, [63]. Each video clip in the dataset includes diverse backgrounds with

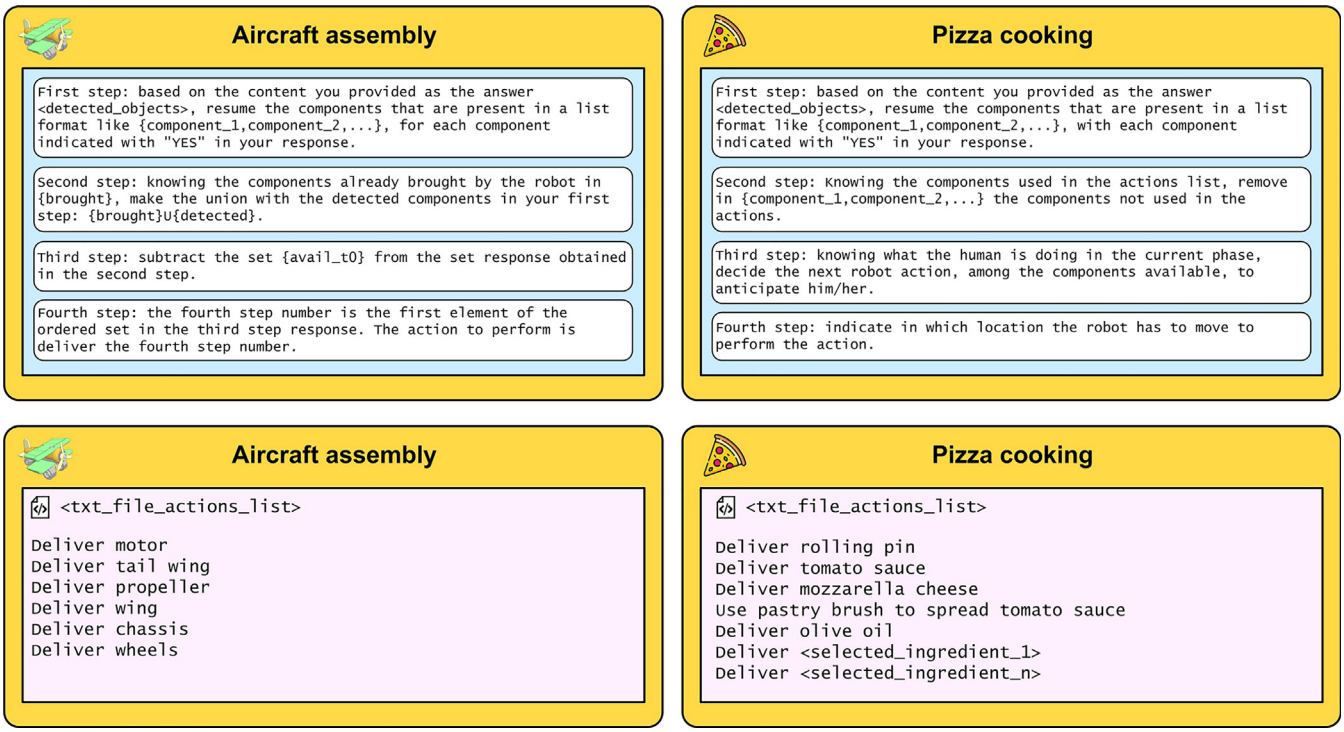

Fig. 9. Reasoning process (light-blue background) divided into different queries for both aircraft assembly and pizza cooking and their respective action lists (pink background) passed in the main R-ManGPT prompt in Fig. 8.

sub-optimal lighting conditions and background noise, enabling the network to learn the relevant hierarchy of visual features that can separate meaningful signals (human motion) from noise (background motion). In this work, the pre-trained weights from the model are fine-tuned to train a custom model capable of recognizing six specific actions: (1) thumb up, (2) roll the dough, (3) spread tomato, (4) spread cheese, (5) putting topping, (6) spread oil.

A specific dataset was collected to fine tune the model for recognizing the six specific actions. For each action, 15 to 30 videos were recorded for the test set and 10 videos for the validation set. Each video is 5 s long and recorded at 16 FPS. To ensure better robustness, the videos were recorded in diverse settings and backgrounds, featuring different subjects.

The model predicts each action at 4 FPS, providing an output classification every 4 frames. To improve stability, an action is confirmed as recognized only after it has been detected with the same recognition output 10 consecutive times. This means the minimum time required for recognition is $10 \times 1/4 = 2.5$ s. However, this minimum time is heavily influenced by the computational resources of the computer used, particularly the availability of a CUDA-enabled GPU. When tested on a standard laptop without GPU acceleration, the model achieved a prediction rate of 2.8 FPS, resulting in a minimum recognition time of approximately 3.57 s, having as input 16 FPS video.

Fig. 10 shows the architecture of the fine-tuned network. At each step, four new frames are input into the network. A 3D CNN extracts the primary spatiotemporal features from these frames, generating a feature vector. This vector is then passed to the LSTM layer, which, after processing a history of four extracted feature vectors, provides an output classification. An accuracy of 100% was obtained for the testing data, which exhibited distinct patterns without occlusions present.

2.4. Experiments and evaluation

2.4.1. System setup

The human operator is working in front of a collaborative robot (5e Universal Robot for aircraft assembly and LBR iiwa Kuka for pizza cooking), Fig. 1. In the aircraft assembly, the storage area is also easily accessible to the human, who can choose to take any component manually. For environment perception during aircraft assembly, two external vision sensors (D435i, Intel RealSense, USA) are used to capture top and side views of the assembly area, Fig. 2. In the pizza cooking scenario, one D435i camera is mounted on the robot flange to capture images of the selected ingredients (by pointing) and detect the storage area. Additionally, a standard webcam is mounted on the side of the workbench to recognize human actions, 4, and a computer microphone is used to receive vocal commands. D-RMGPT communicates with the robot via TCP-IP sockets.

2.4.2. Evaluation

To evaluate the D-RMGPT framework, a series of experimental tests were conducted, focusing on the system robustness and effectiveness in assisting operators during task execution. These tests were carried out under laboratory conditions, encompassing diverse scenarios to thoroughly assess the framework's flexibility, accuracy, and responsiveness.

For the toy aircraft assembly, three types of experiments (#1, #2 and #3) were conducted to evaluate the system's robustness in detecting specific components within the assembly scenario, as well as its flexibility in adapting to operator decisions, effectively assisting both inexperienced and experienced users. For the pizza cooking we evaluated variability by testing all possible ingredient combinations, experiment #4. This scenario aimed to assess the framework's ability to handle multimodal inputs, facilitate easy process customization, and manage tasks of varying complexity. Furthermore, the system's capability to anticipate the next robot

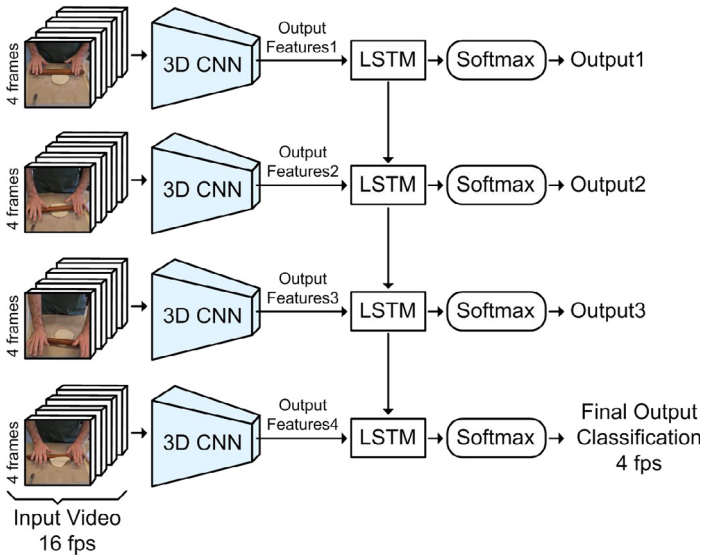

Fig. 10. 3D CNN-LSTM network architecture for action recognition.

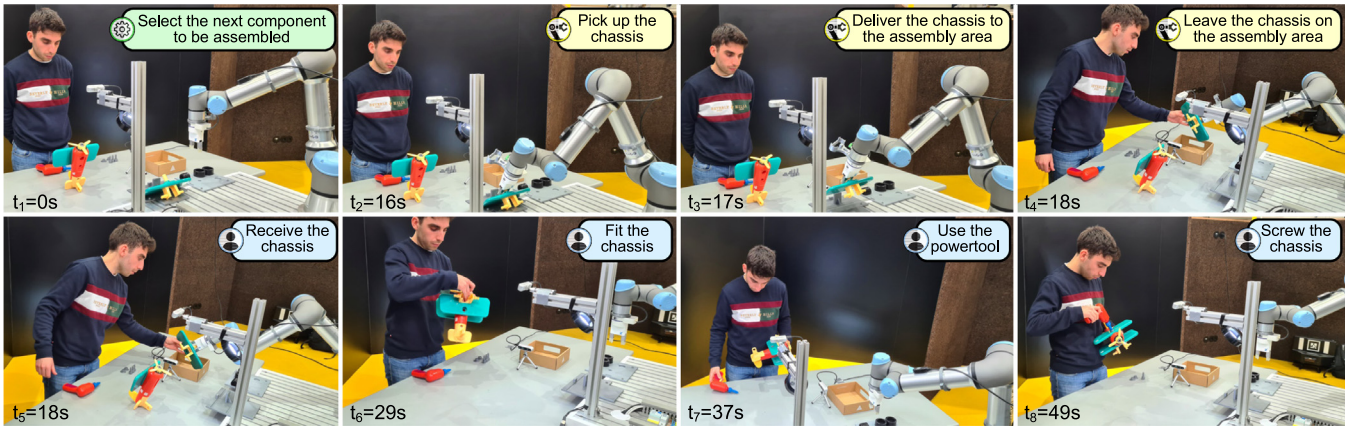

Fig. 11. Assembly step of the toy aircraft chassis assisted by D-RMGPT.

action was evaluated considering both the fluidity and speed of execution, experiment #5.

In experiment #1, the performance of D-RMGPT was evaluated through a series of 12 tests, each conducted by a different inexperienced operator with no prior knowledge of a feasible assembly sequence for the toy aircraft. D-RMGPT guided both the operator's and the robot's assembly actions, assisting in component delivery, using the robot, and following a feasible assembly order. However, it did not provide explicit instructions on how to assemble the components, leaving that decision to the operator's intuition. This approach was consistently applied across all tests of the framework. For each test, we evaluated the number of false positives and false negatives during the detection phase, the total time required to complete the assembly, the average GPT-4V processing time, and whether the aircraft was successfully assembled.

In experiment #2, an experienced operator was assisted by D-RMGPT to evaluate the system's flexibility in a set of 3 tests. At a given assembly step, the operator did not follow the D-RMGPT assembly sequence recommendations, allowing to evaluate how the system reacts and recover from such scenarios.

In experiment #3, the assembly process guided by D-RMGPT was performed by 10 different inexperienced operators, without any prior knowledge of the assembly process or previous explanation. Then, another 10 inexperienced operators performed the same assembly process manually, without being guided by D-RMGPT and without robot assistance. The toy aircraft instructions manual for assembly was provided to them 1 min before starting the assembly. The time taken to complete the assembly was recorded to compare the two testing scenarios.

In experiment #4, focused on pizza cooking, all 15 possible ingredient combinations derived from 4 available ingredients were tested. The trials began with configurations using a single selected ingredient and progressed to recipes that included all four toppings.

In experiment #5 we evaluated scenarios where the operator intentionally chose a different ingredient from the one suggested by the robot, thereby modifying the process flow.

2.4.3. Baseline comparison

The component detection module DetGPT-V was compared with two state-of-the-art VLM-based object detectors, ViLD and OWL-ViT. They have been used in popular foundation models robotics applications such as SayCan, SocraticModels and Grounding Detection. Precision and recall are calculated for each model. To make the comparison fair, since GPT-4V is used to identify the

Table 1
Experiment #1 performance results of assembling the toy aircraft assisted by D-RMGPT in a set of 12 tests, each performed by a different inexperienced operator.

Test	Nº FP	Nº FN	Total Time (s)	Average GPT Time (s)	Success
1	1×Tail Wing	0	317	11	✓
2	0	0	345	13	✓
3	0	0	303	15	✓
4	0	1×Tail Wing	291	16	✓
5	0	0	309	10	✓
6	0	0	263	11	✓
7	0	2×Propeller	289	13	✓
8	0	1×Wheel	319	11	✓
9	0	0	346	19	✓
10	0	2×Tail Wing	326	15	✓
11	2×Chassis	2×Tail Wing	-	19	✗
12	2×Chassis	2×Tail Wing	-	19	✗

Table 2
Experiment #2 results of assembling the toy aircraft guided by D-RMGPT for an experienced operator that did not follow the D-RMGPT recommendations (FP means False Positive, FN means False Negative).

Test	N° FP	N° FN	Tot. time (s)	Av. time (s)	Success	Assembly order
1	1×Chassis	1×Motor 1×Propeller 3×Tail Wing	252	14	✓	1-2-3-4-8-5-6-7-9
2	0	1×Motor	273	15	✓	1-2-3-4-6-7-8-5-9
3	0	0	250	13	✓	1-2-3-4-8-6-5-7-9

presence of the aircraft components but not its position in the image, ViLD and OWL-ViT are also used only as detectors. A detected component is considered True Positive if it present in the image and it is identified by ViLD or OWL-ViT, even if the bounding box is incorrect (IoU=0%). The comparison was conducted exclusively for toy aircraft components, whose detection is more challenging.

3. Results and discussion

In experiment #1, for a set of 12 tests, each conducted by a different inexperienced operator, D-RMGPT assisted and guided the operators to complete the assembly of the toy aircraft, Fig. 11. These operators did not know a feasible assembly sequence beforehand. Table 1 shows the performance values, including false positives and false negatives, calculated by summing the number of incorrect component detections from all the images analyzed during each test. When a component detection is incorrect, the wrong component is indicated in Table 1. The total time required to complete the assembly in the first 10 tests was, on average, approximately 311 s. The variability is mainly due to differences in the time required to process information on the OpenAI servers. This is the time D-RMGPT requires at each assembly step to process images and decide the next component to recommend for assembly. The average GPT processing time for each assembly test is shown in Table 1. Since each assembly involves 7 main calls to OpenAI servers, this processing time accounts for about 30% of the total assembly time. The most time-consuming stage is at the DetGPT-V module, which requires substantial processing due to the volume of information sent to the server and the need for specific component detection. Before obtaining the processing times reported in this work, we carried out a prompt-optimization phase that improved both the accuracy and reliability of the output and the time needed to produce it. This optimization reduced latency by roughly 40%. The Appendix presents the tested prompts and the optimization steps that led to the final version shown in Fig. 6. Splitting the detection tasks into separate, targeted queries significantly reduced response times.

As shown in Table 1, out of 12 tests, 10 resulted in successfully completing the assembly task. In the last two tests, the operator was not able to finish the assembly because the system mistakenly recognized the chassis as already installed when it was not. Consequently, the robot never delivered the chassis, which also prevented the installation of subsequent components that required the chassis to be previously assembled. While existing false positives and negatives did not always compromise the assembly, their impact depended on the component. In tests 1, 4, 7 and 8 the system was able to recover from incorrect component detections.

Fig. 12 shows an example assembly sequence suggested by D-RMGPT, performed by an inexperienced operator. This example demonstrates that even in the presence of false positives and negatives the system was still able to guide the operator to successfully assemble the toy aircraft. Fig. 13 shows the assembly sequence steps suggested by D-RMGPT in case of a detection failure. A false positive detection of the chassis led to the inability to complete the assembly. The system cannot complete the assembly process if a component is falsely detected as present (false positive) when it is not, particularly if that component is a prerequisite for the next assembly step. In such cases, the robot delivers the subsequent component for assembly, but the operator is unable to proceed because the required precedence relationships have not been satisfied.

In experiment #2, the D-RMGPT's flexibility in adapting to scenarios where an experienced operator does not follow the assembly recommendations was evaluated. In a set of 3 tests, the operator assembled some of the suggested components while also picking up other components from the storage area, disregarding the suggested ones. The first four components are mandatory to be assembled in order, otherwise, the subsequent components cannot be installed. From the fifth component onward, the order can vary slightly, always respecting the assembly constraints. Table 2 shows the results, demonstrating the system's ability to adapt and guide the assembly process under these conditions. Fig. 14 details the assembly steps of test 1, in which after assembling component #4, the system recommended assembling component #5. However, the operator did not follow the D-RMGPT recommendation and instead assembled component #8. In this scenario, D-RMGPT managed to successfully complete the assembly proposing the sequence 1-2-3-4-8-5-6-7-9. The system guides the operator while maintaining flexibility by detecting only the components that have already been installed and keeping track of those delivered in previous steps. Although the system suggests an optimal component for assembly based on the current state, the operator is not obligated to follow this recommendation and can freely adjust the assembly strategy when possible. Considering the results obtained in experiment #1 and #2, it is possible to

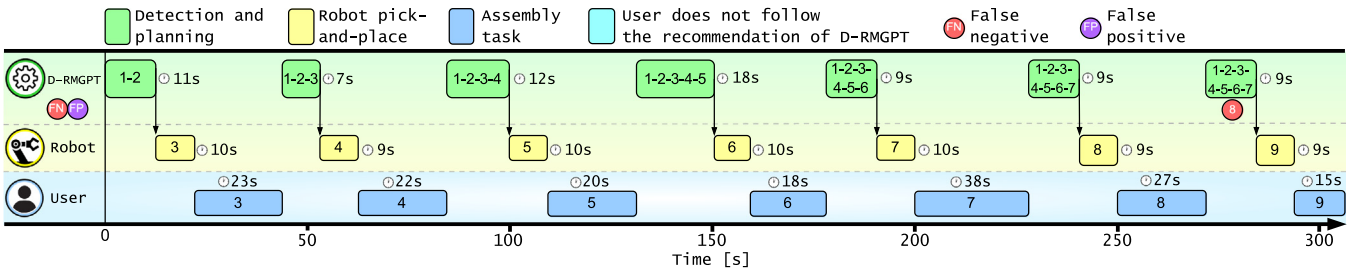

Fig. 12. Experiment #1 assembly sequence steps suggested by D-RMGPT and performed by an inexperienced operator. The assembly was successfully achieved despite the presence of false positives and negatives. The numbers inside the boxes represent the component numbers.

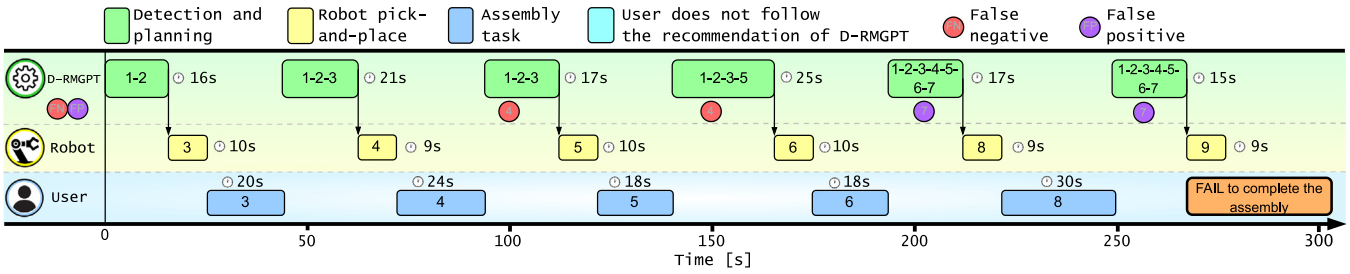

Fig. 13. Experiment #1 assembly sequence steps suggested by D-RMGPT, performed by an inexperienced operator, where a false positive detection of the chassis (component #7) led to the inability to complete the assembly. The numbers inside the boxes represent the component numbers.

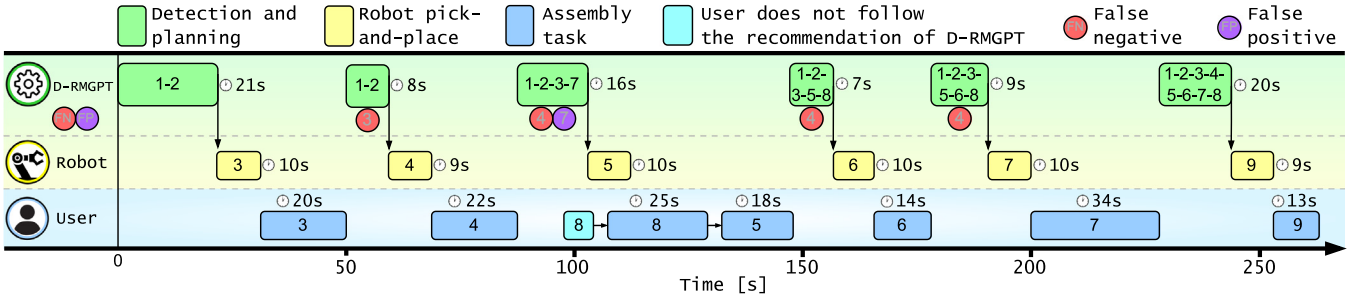

Fig. 14. Experiment #2 assembly sequence steps performed by an experienced operator that did not follow the D-RMGPT recommendation in a single assembly step. D-RMGPT managed to successfully complete the assembly. The numbers inside the boxes represent the component numbers.

conclude that success is obtained in 86.6% of cases with a Wilson 95% confidence interval of [62.7%, 96.6%].

In experiment #3, we conducted another set of tests to compare the time required to complete the assembly between two groups of 10 inexperienced operators. The first group was assisted by D-RMGPT, while the second group performed the assembly manually without any guidance from D-RMGPT or robot assistance. The first 10 operators had no prior knowledge of the assembly object or previous explanation about it. For the second group of 10 operators, they performed the assembly manually, without guidance from D-RMGPT or robot assistance, and were provided with the toy aircraft assembly manual of instructions only 1 min before starting the assembly. The total assembly time for each operator is shown in Fig. 15, highlighting the average value and the standard deviation for each group. Results show that the average assembly time for the first group, assisted by D-RMGPT, is 311 s, which is approximately 33% lower than the average time required for the second group (467 s). The standard deviation values for the first group ($\sigma = 26$ s) are smaller than for the second group ($\sigma = 89$ s), indicating that when the assembly is assisted and guided by D-RMGPT, there are less variability, making the process more uniform and predictable, and less user-dependent. In the second group, some operators assembled components in the wrong sequence and had to disassemble and reassemble them, leading to increased variability in assembly time. The hypothesis $H_0 : \mu_{ass} = \mu_{man}$ versus $H_1 : \mu_{ass} < \mu_{man}$ have been tested via a Welch t-test after confirming approximate normality of both groups with the Shapiro–Wilk test and unequal variances with Levene's test. The analysis yielded a Welch-t statistics of -5.335 and $p = 0.000139$, providing strong evidence to reject the null hypothesis. These results indicate that the assisted method significantly reduces completion time compared to the manual method (see Table 3).

The difference between assisted and manual assembly times was quantified using Hedges'g to account for small sample sizes. The effect size was extremely large, $g = -2.29$, indicating that assisted assembly was more than two standard deviations faster than manual assembly.

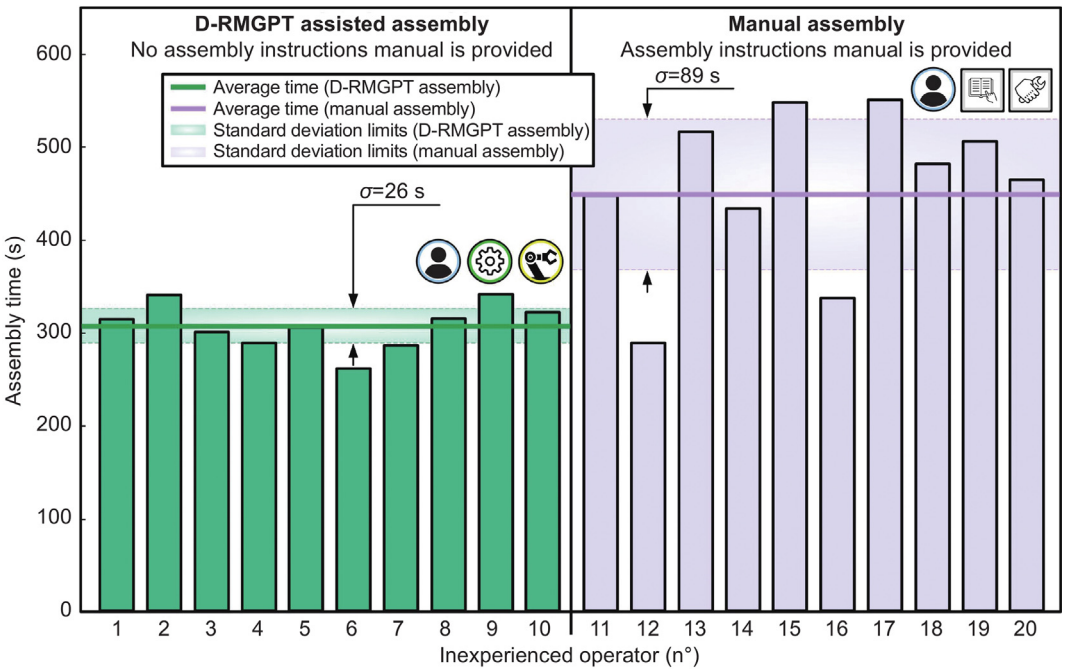

Fig. 15. Experiment #3 assembly time of a first group of 10 inexperienced operators assisted by D-RMGPT (green), and a second group of 10 inexperienced operators that performed the assembly manually without any guidance from D-RMGPT or robot assistance (purple).

Table 3
Statistical summary of completion times for 10 trials of Experiment #3 featuring different metrics for assisted and manual assembly.

Metric	Assisted	Manual
Mean time (s)	310.8	466.8
Standard deviation (SD) (s)	25.74	88.80
Median (s)	313	484
First quartile (Q1) (s)	294.0	444.25
Third quartile (Q3) (s)	324.25	525.0
Interquartile range (IQR) (s)	30.25	80.75
95% CI of the mean (t-based) (s)	[292.39, 329.21]	[403.28, 530.32]
95% CI of the mean (bootstrap) (s)	[295.3, 325.4]	[411.40, 514.40]

Table 4
Component detection baseline comparison between DetGPT-V (based on GPT-4V), ViLD and OWL-ViT. Precision (P) and recall (R) are calculated for each model in a set of 15 tests for each component.

Component	OWL-ViT		ViLD		DetGPT-V	
	P	R	P	R	P	R
Fuselage	1	1	1	1	1	1
Motor	1	0.83	N/A	0	1	0.97
Tail Wing	0.80	0.80	1	0.60	0.98	0.87
Propeller	0.57	1	0.75	0.75	1	0.95
Wing	1	1	0.33	0.75	1	1
Chassis	0.50	0.50	1	0.50	0.85	1
Wheel	0.25	1	0.50	1	1	0.93

In summary, for inexperienced operators, using D-RMGPT reduces both the average assembly time and the variability of the process.

Finally, the component detection module DetGPT-V was compared with state-of-the-art VLM-based object detectors, ViLD and OWL-ViT. For each component, the precision $P = \frac{TP}{TP+FP}$ and recall $R = \frac{TP}{TP+FN}$ were calculated in a set of 15 tests. As shown in Table 4, DetGPT-V offers significantly more accurate detection performance than the other two detectors. For example, in the case of component #2 (upper fuselage), there is a yellow tail wing on the orange main body, Fig. 3. OWL-ViT and ViLD detectors mistakenly identify this part as a second yellow tail wing component, while DetGPT-V can correctly recognize that it belongs to the upper fuselage component. A common mistake made by these open-vocabulary detectors is that they confuse components of similar shape and color, such as mistaking component #3 (motor) with component #8 (wheels). Overall, for this type of application, DetGPT-V, based on GPT-4V, performs better than state-of-the-art VLM-based object detectors like ViLD and OWL-ViT. In scenarios where understanding the current scene is critical before planning the robot's next action, the DetGPT-V module represents a significant advancement. It enhances component detection performance and accommodates a wider variety of objects, thereby expanding the range of applicable scenarios, including highly customized environments with specific components. Recognition accuracy metrics for DetGPT-V are calculated based on the results from the 12 tests shown in Table 1 and the 3 tests in Table 2.

In Experiment #4, we collected data from 15 tests, covering all possible recipe combinations using 4 topping ingredients, Fig. 16. The key features of the tests and results are summarized in Table 5. Each test starts with the operator selecting the type of recipe to prepare. A green check mark indicates that the system successfully recognized the operator's decision, either through speech commands specifying the number of toppings (N° Top.), or through image recognition identifying the selected toppings (Top. Sel.). The available toppings are the salami (S), mushrooms (M), olives (O), or basil (B). A red cross denotes instances where the system failed to accurately perceive the operator's input.

The average time for image detection (AT Det.) refers to the time the system requires to process image inputs, specifically the images of the pointing finger and the images of the ingredients and kitchen utensils present in the storage area. A green check mark indicates correct detection of every component at each stage, while a red cross indicates instances where detection failed. Action recognition false positives (A.R. FP) and false negatives (A.R. FN) are recorded, listing all actions incorrectly recognized by the system during the action recognition phase. The average

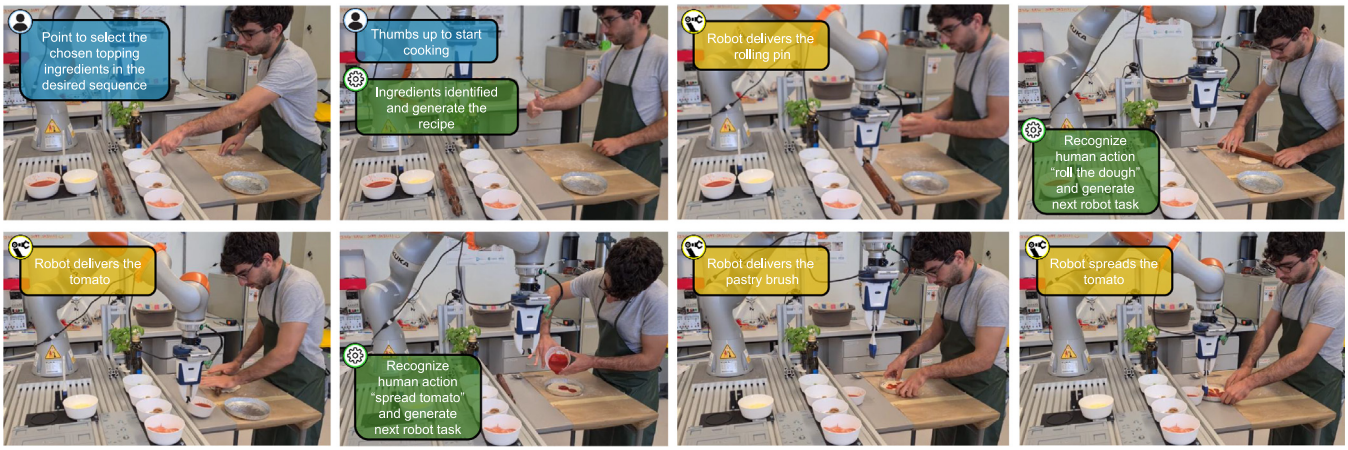

Fig. 16. Experiment #4: cooking scenario featuring the initial steps of pizza preparation assisted by the D-RMGPT that guides the cooking process.

Table 5
Results of Experiment #4 focused on the pizza cooking process carried out by an inexperienced operator.

Test	Nº Top.	Top. Sel.	AT Det. (s)	A.R. FP	A.R. FN	AT A.R. (s)	R. Act.	AT R.D. (s)
1	1 ✓	S ✓	4.93 ✓	/	/	6.24	7 ✓	1.50
2	1 ✓	B ✓	4.18 ✓	/	/	6.55	7 ✓	0.74
3	1 ✓	M ✓	4.95 ✓	/	/	6.29	7 ✓	1.51
4	1 ✓	O ✓	4.15 ✓	/	Oil	N/A	✗	N/A
5	2 ✓	S,B ✓	4.13 ✓	/	/	6.02	8 ✓	0.80
6	2 ✓	S,M ✓	4.39 ✓	/	/	6.51	8 ✓	0.86
7	2 ✓	S,O ✓	4.96 ✓	/	Oil	N/A	✗	N/A
8	2 ✓	M,B ✓	5.93 ✓	/	/	5.85	8 ✓	0.76
9	2 ✓	O,B ✓	5.27 ✓	Topping	Oil	6.99	8 ✓	0.80
10	2 ✓	M,O ✓	4.86 ✓	/	/	5.59	8 ✓	1.16
11	3 ✓	S,M,B ✓	4.22 ✓	/	/	5.41	9 ✓	0.75
12	3 ✓	S,O,B ✓	4.82 ✓	/	/	5.67	9 ✓	0.76
13	3 ✓	S,M,O ✓	4.24 ✓	Topping	Oil	7.10	9 ✓	1.03
14	3 ✓	M,O,B ✓	4.25 ✓	/	/	5.71	9 ✓	1.32
15	4 ✓	S,M,O,B ✓	4.85 ✓	/	/	5.65	10 ✓	0.93

time the system takes to recognize an action (AT A.R.) is the average time required to reach a stable classification, defined as obtaining the same classification result at least ten consecutive times. False negatives also include actions that the system was unable to recognize in a stable manner, meaning it could not meet the stability criterion and consistent classification results within a specified maximum time limit of 15 s. In these cases, the system failed to maintain a consistent recognition output long enough to satisfy the stability requirement. The number of actions performed by the robot (R. Act.), along with a green check mark, indicates that the system correctly identified and executed the appropriate action to assist the human, leading to the successful completion of the task. In contrast, a red cross indicates a failure in the assistance task, resulting in the task's incompletion. Finally, the average time for robot decision-making (AT R.D.) represents the average time the system required to make a decision within the robot assistance module (R-ManGPT).

Table 5 shows that out of the 15 tests, only two failed to complete the pizza cooking process. Specifically, failures occurred in tests 4 and 7 due to errors in action recognition. In these cases, the final action of *spread oil* was not recognized within the maximum allowable time of 15 s, causing the system to register a failure. As observed across the 15 tests, the action class most prone to error was *spread oil*. In tests 9 and 13, although the *spread oil* action was not recognized (false negative), the system incorrectly identified the action *putting topping* as occurring (false positive for the *putting topping* class). Despite these inaccuracies, the R-ManGPT system successfully guided the operator through the task, ultimately achieving successful completion.

Table 6
Experiment #4 average times of each phase of the pizza cooking process across all the 15 tests.

Process phase	Average time (s)
Utensils and/or ingredients detection	4.67
Robot decision	0.97
Thumb up	6.31
Roll the dough	6.18
Spread tomato	4.97
Spread cheese	4.76
Putting topping	6.47
Spread Oil	7.30

Table 6 presents the average times recorded at key phases of the cooking process, averaged over the 15 tests. The *spread oil* action had the longest average recognition time (7.30 s), while the *spread tomato* action had the shortest, averaging 4.97 s. The total duration of each test also depended on the operator's skill in executing the tasks. In Experiment #5, as in Experiment #2, the framework's ability to assist experienced operators was tested. These operators did not follow the suggested sequence of ingredient delivery, instead modifying the order or using two ingredients simultaneously. For example, while the robot delivered one ingredient, the operator would retrieve a different one from the storage area and proceed to use it. Eight tests were conducted to evaluate the framework's ability to adapt to operator decisions. As shown in Table 7, among the eight tests performed, only Test 2 resulted in a failure. This failure was caused by the system's

Table 7
Experiment #5 results for assisting operators who did not follow the suggested sequence of ingredient delivery.

Test	Nº Top.	Top. Sel.	AT Det. (s)	A.R. FP	A.R. FN	AT A.R. (s)	R. Act.	AT R.D. (s)
1	1 ✓	S ✓	3.37 ✓	/	/	7.59	5 ✓	1.05
2	1 ✓	M ✓	3.87 ✗	/	Roll	N/A	✗	N/A
3	1 ✓	B ✓	4.25 ✓	/	/	6.57	5 ✓	0.62
4	1 ✓	O ✓	4.10 ✓	/	/	6.16	5 ✓	0.67
5	4 ✓	S,M,O,B ✓	4.08 ✓	/	/	5.67	8 ✓	0.72
6	4 ✓	S,M,O,B ✓	4.37 ✓	/	/	6.60	8 ✓	0.76
7	4 ✓	S,M,O,B ✓	5.32 ✓	/	/	5.95	8 ✓	0.96
8	4 ✓	S,M,O,B ✓	4.79 ✓	/	/	5.46	8 ✓	0.74

Table 8
Experiment #5 sequences of kitchen utensils and/or ingredients used by operators who did not follow the suggested order.

Test	Kitchen utensils and/or ingredients used in an unplanned sequence
1	Rolling pin, tomato, mozzarella, salami
2	Test failure
3	Brush, mozzarella, mushrooms, oil
4	Tomato, brush, olives, oil
5	Salami, mushrooms, olives, basil
6	Salami, oil, olives, mushrooms
7	Salami, oil, olives, basil
8	Tomato, rolling pin, salami, olives

inability to recognize the *roll the dough* action. Since the robot was unaware of the operator's activity, it was unable to provide the necessary support to complete the task. Additionally, during this test, an incorrect detection occurred as the pastry brush was not recognized as present. In all other tests, the system successfully identified actions, components, and human inputs, consistently suggesting appropriate robotic responses. It effectively guided the operator through the task, even when they deviated from the suggested order and took their own initiative. Table 8 reports the kitchen utensils and ingredients used by the operator in a customized sequence.

Overall, across Experiment #4 and Experiment #5, the tests demonstrated a success rate of 87% with a Wilson 95% confidence interval of [67.87%,95.46%]. Even recipes requiring a higher number of robot-assisted tasks, up to 9 or 10 steps, were successfully completed. The framework consistently selected the correct actions to support the operator, even during longer and more complex tasks involving multiple action options. The multimodal detection system also proved highly reliable, achieving a 98.5% success rate in correctly identifying inputs. Both speech inputs (for the number of toppings) and image inputs (for topping selection and ingredient detection) were recognized with high accuracy. The introduction of action recognition through a 3D-CNN LSTM gives the robot an anticipatory capability that speeds up the assembly task. At the same time, it can become a source of failure in the assistive process. In experiments #4 and #5, three failures occurred due to false negatives in action recognition. When the downstream module, in this case R-ManGPT, does not receive the required input, the process fails. In both the cooking pizza and toy assembly scenarios, the subprocesses (3D-CNN LSTM, DetGPT-V, R-ManGPT) operate in sequence, which makes a continuous flow of input essential. Instead, their decoupled design increases system robustness. Even when a subprocess receives incorrect input, such as a false positive, the downstream module can still produce a correct result, allowing the task to succeed.

3.1. Discussion

The tests conducted across both application scenarios highlight that, while LMMs may occasionally produce hallucinations by incorrectly identifying the presence of components, they consistently demonstrate strong performance in robot task planning. As illustrated in Tables 1, 2, and 5, even when such hallucinations occur, the system remains capable of completing the task. Not all errors critically impact task execution, as the system often recovers from mistakes in subsequent iteration steps, allowing it to continue progressing toward the objective.

The feasibility, usability and effectiveness of the proposed framework have been tested and demonstrated in two real-world application scenarios designed to showcase the system's detection, perception, and planning capabilities, as well as its scalability across different contexts while maintaining its core objective of assisting and guiding the operator through the process. The system architecture supports both reactive behaviors, responding to the operator's actions (as in the toy aircraft assembly), and proactive behavior, anticipating the operator's next actions (as in the pizza cooking scenario).

The toy aircraft assembly scenario highlights the system's ability to detect specific components within a scene using tailored image prompts. DetGPT-V introduces a reliable component detection framework by leveraging an innovative prompt engineering approach, achieving these results without the need for specific training. This provides the system with significant flexibility and adaptability across various applications. The cooking scenario showcase the system's ability to assist operators in non-standard, customizable processes tailored to their needs. It accommodates workflows ranging from simple to more elaborate tasks. Additionally, it offers the possibility of process customization through intuitive and multimodal interfacing, such as speech commands or captured images, allowing operators to directly express their preferences.

Beyond the two application scenarios, it is useful to contextualize D-RMGPT with respect to recent LLM-based robotic frameworks such as SayCan [44], Socratic Models [42], and RoboGPT [64]. These approaches share the goal of linking high-level reasoning with embodied execution, yet they differ substantially from our framework in terms of perception–planning integration, human–robot collaboration, and deployment overhead. SayCan relies on an LLM constrained by RL-trained value functions, enabling feasibility-aware planning but requiring extensive task-specific reinforcement learning and offering limited adaptability to human actions. Socratic Models coordinate separate vision and language models through natural-language exchanges, which makes them dependent on external detectors that often struggle with fine-grained object recognition. This is an issue addressed in our work through the unified GPT-4V-based DetGPT-V perception pipeline. RoboGPT employs a fine-tuned LLaMA planner trained on a large 67k-sample dataset, combined with predefined skills and map-based re-planning, but it targets autonomous instruction following and demands significant data curation. In comparison, D-RMGPT introduces four key novelties. First, it unifies perception and reasoning within a single multimodal model (GPT-4V), eliminating the need for external object detectors, bounding-box pipelines, or large fine-tuning datasets, while achieving superior detection performance in structured assembly. Second, it supports continuous, image-grounded re-planning driven by precedence relationships or operator actions,

allowing the system to adapt to human deviations, a capability largely absent in SayCan and RoboGPT. Third, D-RMGPT is explicitly designed for human-in-the-loop collaborative work, integrating speech, gesture, pointing and action classification via a lightweight 3D CNN-LSTM module to anticipate operator needs and adjust robot actions online. Fourth, its zero-training, plug-and-play nature makes it inherently flexible and transferable across domains as the same architecture successfully handled a constrained assembly task and an open-ended cooking scenario with high success rates. Taken together, these elements position D-RMGPT as a novel, unified multimodal planning framework that not only simplifies system integration compared to existing LLM-robotics approaches but also enables robust and intuitive human–robot collaboration without task-specific training data or manually engineered skill hierarchies. This represents a meaningful step toward practical, adaptable, and generalizable collaborative robotic systems. Results were obtained applying a prompting strategy that avoids ambiguity, given the high level of specificity required for component detection (in the assembly task) and for action planning (in cooking pizza). As in the Appendix, less detailed and less structured prompts led to lower accuracy. That said, the use of more flexible prompts for more generic, domain-familiar tasks remains an open direction for future studies. The proposed framework also mitigates component occlusion by allowing multiple images as input, ensuring that all components that need to be recognized are visible to the system.

An intermediate module could be introduced between each modules to verify the feasibility of the LMM outputs or compensate for the absence of input. While this solution would likely improve the task completion success rate, it would also increase processing times, potentially making the applications less fluid and more cumbersome. Therefore, the question of whether there is a tangible advantage in implementing more elaborate reasoning verification techniques and incorporating interactive feedback with the system remains open for further investigation. As a potential fallback mechanism, while still respecting the assumption of no task-specific training and therefore avoiding data-intensive AI models, we propose a strategy in which the system explicitly verifies the presence of each newly detected component through a dedicated prompt. This additional targeted query minimizes the impact of false positives, which are the primary cause of task failure. The system performs a double-check for every newly identified component, confirming or rejecting its actual presence before proceeding. To compensate for the absence of external input, a future extension of this approach could allow the system to infer the operator's ongoing action through a tailored prompt that leverages the temporal evolution of the task. This would enable the system to maintain progress without triggering unnecessary stops, as observed in the failure cases of the cooking pizza scenario.

To address latency, future iterations will explore on-device or hybrid inference pipelines that combine lightweight local computation with remote LMM calls when necessary. In parallel, we aim to develop active verification loops, allowing the system re-queries uncertain detections, validate state transitions, and leverages temporal task evolution to avoid unnecessary task interruptions. We will investigate how the number and diversity of input views affect detection robustness, balancing perceptual performance with computational cost. We also plan to integrate mechanisms for learning user preferences and interaction patterns, further improving intuitiveness and personalization in collaborative tasks.

3.1.1. Practical usability

Responsiveness is primarily influenced by multimodal perception and planning inference times. In the aircraft assembly scenario, GPT-based processing accounts for approximately 30% of the total task duration, with an average per-step inference time reported in Table 1. This latency does not directly affect robot motion execution, as D-RMGPT operates at a high-level planning layer and is not involved in low-level control loops. In collaborative tasks, human actions typically require several seconds to complete, making the observed inference times compatible with natural interaction rhythms. Furthermore, a dedicated prompt optimization phase reduced inference latency by approximately 40%, demonstrating that responsiveness can be further improved through prompt design and system tuning.

System stability is ensured through a combination of architectural and interaction design choices. This prevents infeasible or unsafe action sequences from being generated, even in the presence of perception uncertainty. The framework follows a human-in-the-loop paradigm, where the robot assists and anticipates actions without forcing task execution. As a result, failures in perception or action recognition do not propagate into unsafe robot behaviors. Experimental results show that the system can often recover from false positives and false negatives in subsequent iterations, maintaining overall task progress. While occasional perception errors are inherent to current large multimodal models, the proposed framework demonstrates robustness by decoupling perception, planning, and execution modules. Even when incorrect detections occur, downstream planning can compensate in many cases, allowing the task to be completed successfully. Critical failures occur only when essential precedence constraints are violated, a limitation that is explicitly acknowledged and motivates future integration of redundancy and verification mechanisms.

From a deployment perspective, D-RMGPT is designed to be modular and hardware-independent. The same framework was successfully deployed using different collaborative robots and camera configurations without retraining the multimodal models. Additionally, the number and resolution of visual inputs can be adjusted to trade off perception accuracy and inference latency, enabling adaptation to different computational budgets and real-world constraints.

Although the experimental validation focuses on two application scenarios, these were deliberately selected to span markedly different collaborative paradigms. The aircraft assembly represents a highly structured task with strict precedence constraints and purely reactive robot behavior, while the cooking scenario embodies an open-ended, user-driven workflow with multimodal interaction and anticipatory assistance. The fact that both scenarios are addressed using the same D-RMGPT architecture, prompt structure, and off-the-shelf multimodal models, without retraining, demonstrates that the proposed framework is task-agnostic by design. Extending D-RMGPT to additional domains would mainly require defining new component lists and discrete action sets, without modifying the underlying perception and planning pipeline.

4. Conclusion and future work

This paper introduced D-RMGPT, a robot-assisted planner based on LMM to assist inexperienced operators in assembly tasks and cooking. D-RMGPT demonstrated its capability to guide both processes, with the robot and human operator in the loop. While assisting inexperienced operators, the system achieved a success rate of 83% in the assembly tasks and 87% in the pizza cooking tasks, without requiring prior information about possible task sequences. The overall success rate among all the tests

conducted is of 86.8% with a Wilson 95% confidence interval of [73.0%, 94.2%]. The results demonstrated that D-RMGPT reduces assembly time variability for inexperienced operators, with a 33% reduction in the assembly time when compared to manual assembly without any guidance from D-RMGPT or robot assistance. The detection module DetGPT-V, compared favorably with state-of-the-art VLM-based object detectors ViLD and OWL-ViT in our specific case scenario.

The results suggest that D-RMGPT is both robust and flexible. It demonstrates robustness by delivering feasible solutions even in the presence of detection errors. It is flexible because it can recover from scenarios where the operator deviates from the recommendations provided by D-RMGPT. This resilience ensures continuity and efficiency in both processes, even when unexpected actions are taken by the operator. Overall, D-RMGPT is a promising and versatile tool, capable of operating in uncertainty without requiring specific training data (except for action classification, if is needed an anticipatory behavior). The system effectively assisted inexperienced operators demonstrating its potential as a robust and adaptable solution for intuitive human–robot interaction and collaboration. The system was evaluated in two scenarios to highlight its main features and strengths, as discussed in Section 3.1. Nevertheless, it can easily be adapted to different scenarios in total different contexts (here is demonstrated in a industrial assembly scenario or in home care robotics scenario), by adjusting the scenario-specific components of the prompt (Figs. 6, 9).

Future developments will focus on increasing responsiveness and generalization capabilities of the proposed framework. It is planned to introduce adaptive prompting designs that allow the system to rewrite or refine its own prompts based on task context and operator actions, thereby reducing reliance on manual prompt engineering. Building on this, D-RMGPT will be extended with multimodal reasoning capabilities, enabling it to better dynamically infer operator intent, anticipate next actions, and select fallback strategies during ambiguous or error-prone situations.

CRediT authorship contribution statement

Matteo Forlini: Writing – original draft, Software, Methodology, Investigation, Data curation. **Mihail Babcinschi:** Supervision, Software, Resources, Methodology. **Giacomo Palmieri:** Writing – review & editing, Validation, Supervision, Resources, Project administration. **Pedro Neto:** Writing – review & editing, Supervision, Resources, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Portuguese national funds through Fundação para a Ciência e a Tecnologia (UID/00285/2025, LA/P/0112/2020, DOI: 10.54499/UID/00285/2025), and the EuGen program EU4EU.

Appendix A. Prompt design process

Before arriving at the final prompt structure for component detection, several iterations of prompt engineering were conducted to ensure deterministic and accurate results in the detection module. Initially, our approach did not differentiate between the various roles, i.e., system, assistant, and user. Instead, the entire request was placed under the user role as a single query. At this stage, the system is provided with an explanation of the purpose of the collected images and the task it is expected to perform, namely, component detection for the aircraft toy assembly. To support this process, reasoning tips are included, and the system is instructed to perform a double-check to validate the presence of each component. This involves visually identifying the component in the images corresponding to the current assembly step and verifying that all precedence dependencies with other components are satisfied. Additionally, an example of well-executed reasoning is provided to illustrate the expected process and help ensure a deterministic and accurate response, Fig. A.1. However, the unreliability of the results necessitated a change in strategy. The output lacked determinism, and errors introduced by the double-check process led to incorrect detection of all components. To address this issue, a new approach was introduced: in addition to the three previously described images, a fourth image, showing the fully assembled aircraft toy, was added, Fig. A.2. The aim was to compare the image of the current step with the complete assembly to identify which components were already present. However, this method also proved inadequate, as it did not consistently improve detection accuracy. The approach was then completely revised, abandoning the attempt to enforce reasoning based on precedence relationships and instead simplifying the reasoning process, Fig. A.3. The output was restructured in a more schematic format to guide the system's response. Additionally, the use of the three roles, system, assistant, and user, was introduced. In this new setup, the assistant role provides the previous step, while the user role asks the system to identify which component has been added in the current step by comparing it to the previous one. As hallucinations persisted, the approach was further simplified by shifting to a step-by-step evaluation, asking whether each component was visible, under the assumption that only one component could be added at a time before arriving at the final solution, an intermediate version of the system was implemented, in which a single query prompted the identification of all components, requiring a simple YES or NO response for each, Fig. A.4. The final step, which achieved the desired level of reliability, involved implementing targeted questions for each component by dividing the requests into separate queries, Fig. 5. This approach encouraged the model to reason step by step. By providing a schematic output format and asking simple, specific questions in isolated queries, the system achieved a deterministic output with minimal hallucinations. In 15 tests, component detection was performed 7 times per test. For each detection, the presence of 8 components was evaluated, resulting in a total of $15 \times 7 \times 8 = 840$ potential error opportunities. The system produced only 22 detection hallucinations, corresponding to 2.3% of the total. Additionally, with the adoption of the latest prompt structure, the average response time of GPT was significantly improved, reaching 14.2s. In contrast, the initial prompt structure resulted in a considerably higher average response time of 25.7s. This optimization led to a substantial reduction of 44.7% in response time. The DetGPT-V prompt structure was developed using the aircraft toy assembly scenario, which presents significant challenges due to the presence of specific pre-assembled components. A primary limitation of the system lies in accurately identifying both the number of components installed and their relative positions within the image.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.birob.2026.100334.

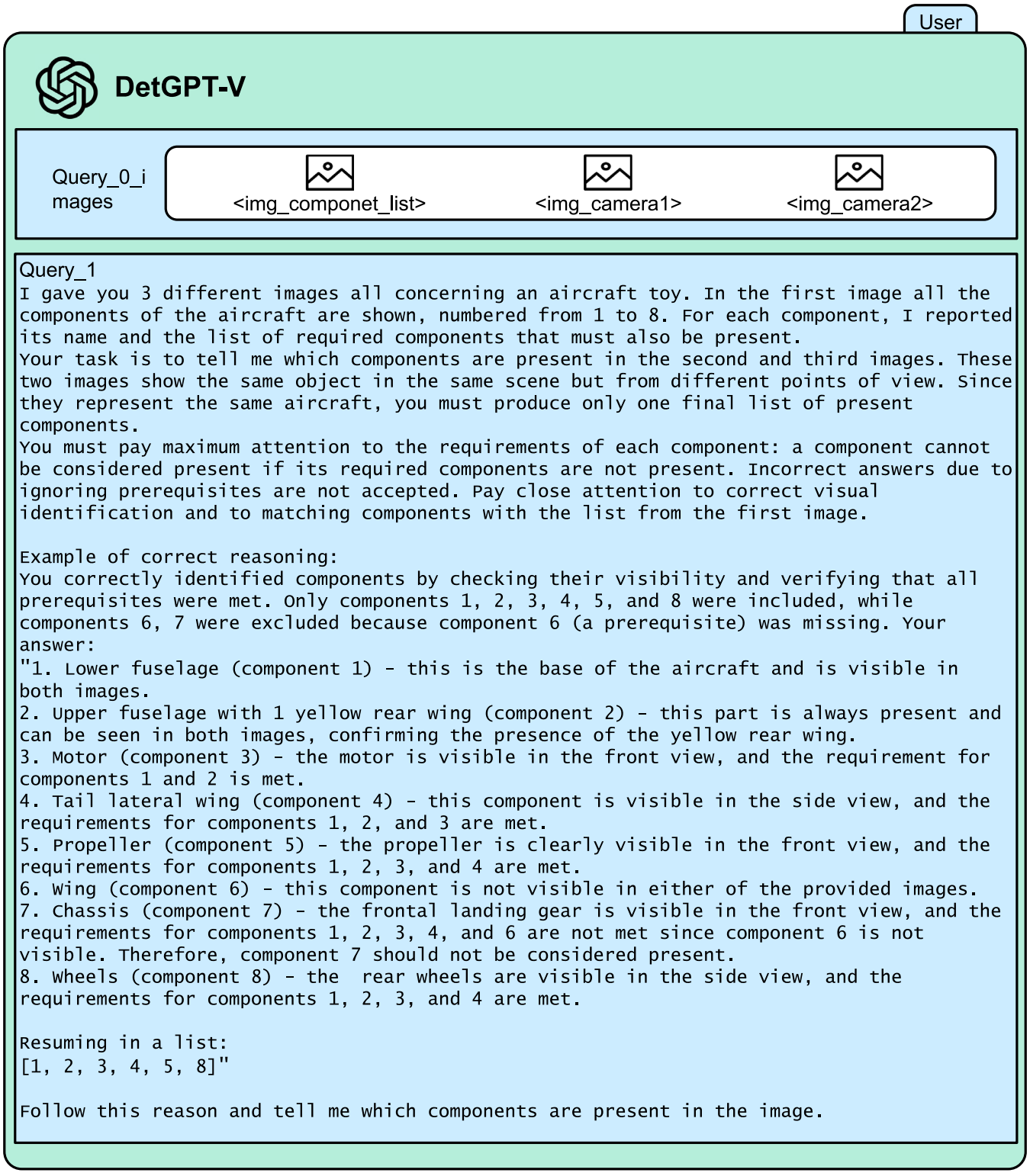

Fig. A.1. Prompt structure for the detection module DetGPT-V featuring three input images.

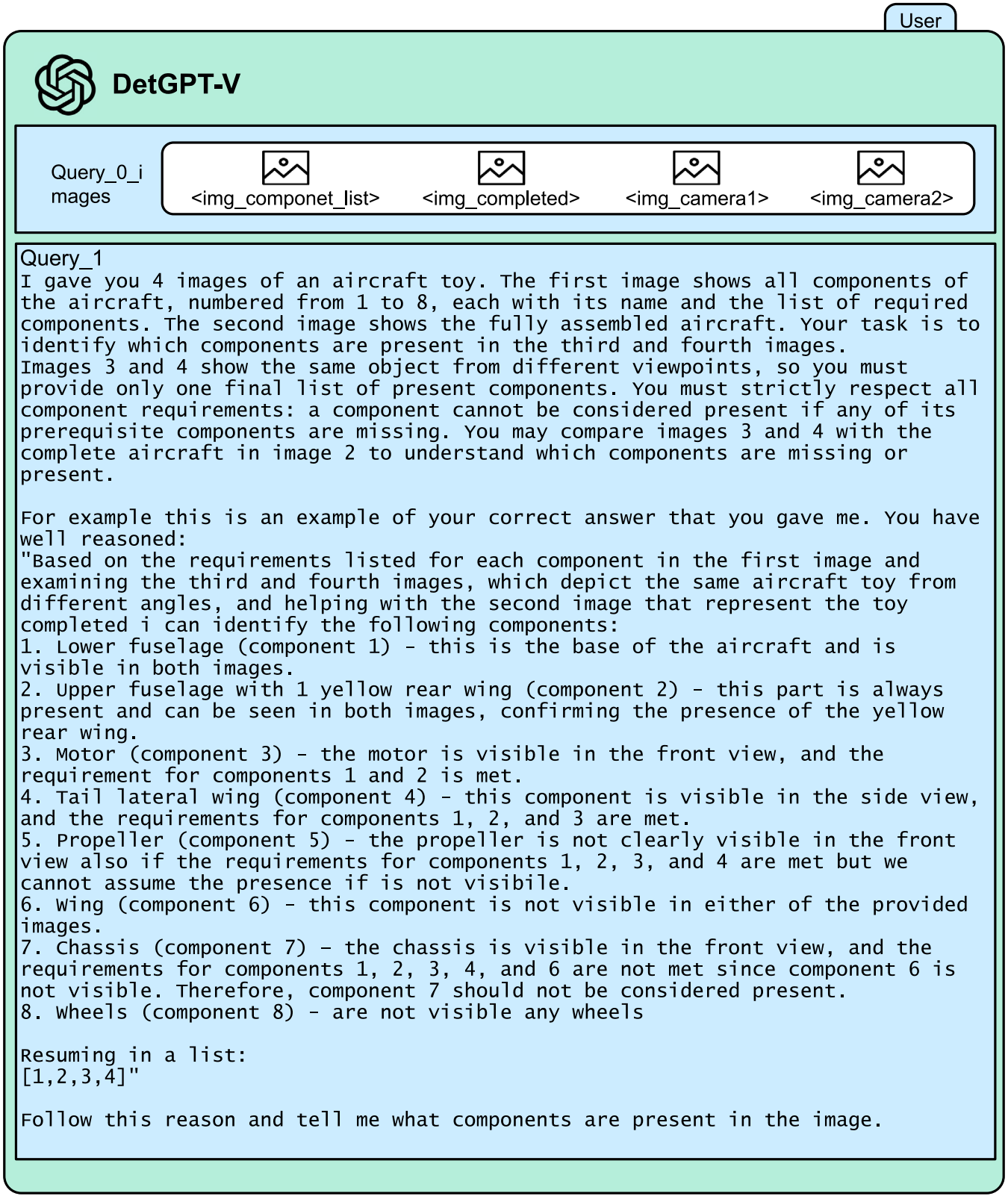

Fig. A.2. Prompt structure for the detection module DetGPT-V featuring four input images.

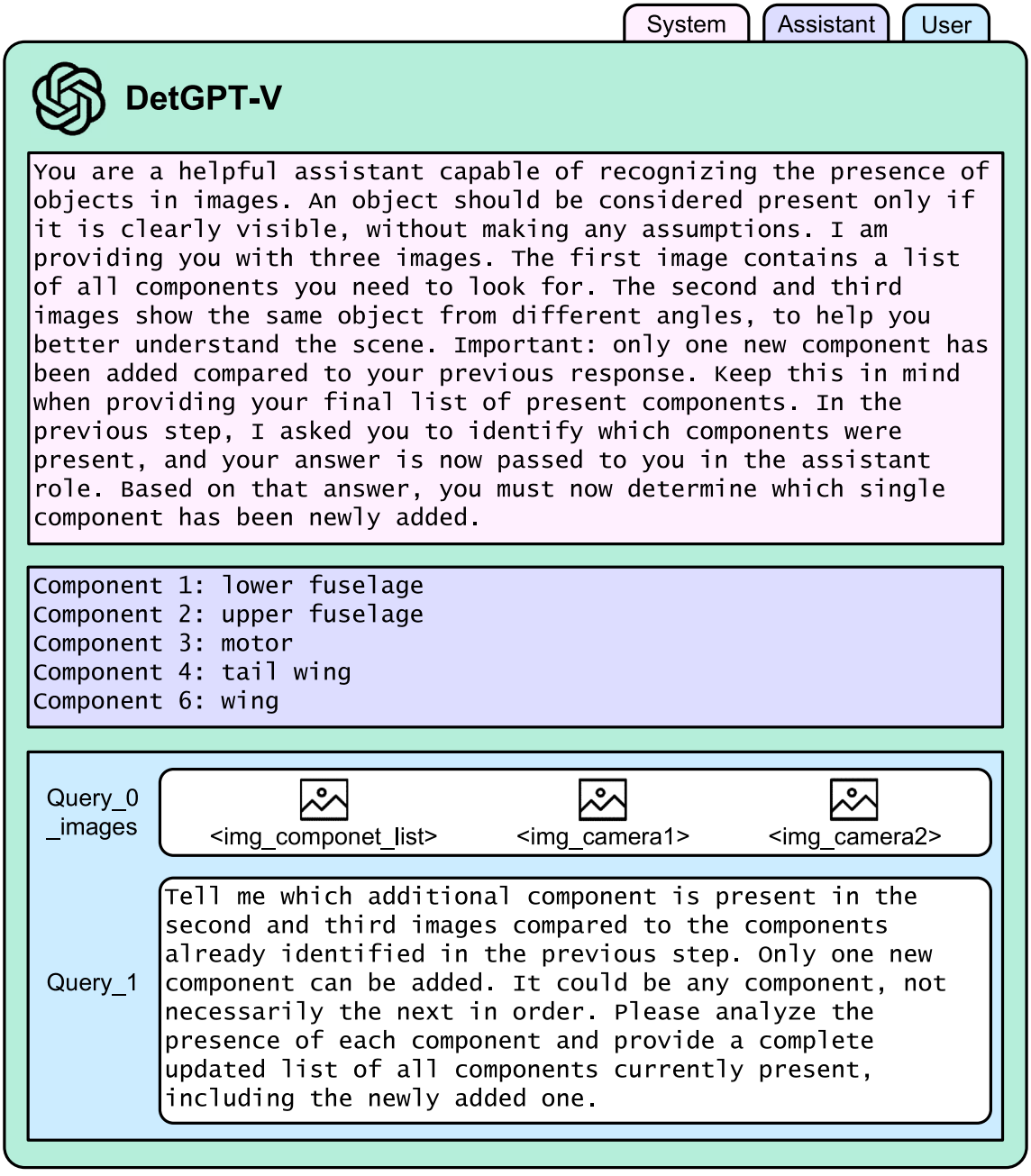

Fig. A.3. Prompt structure for the detection module DetGPT-V featuring three roles: system, assistant and user.

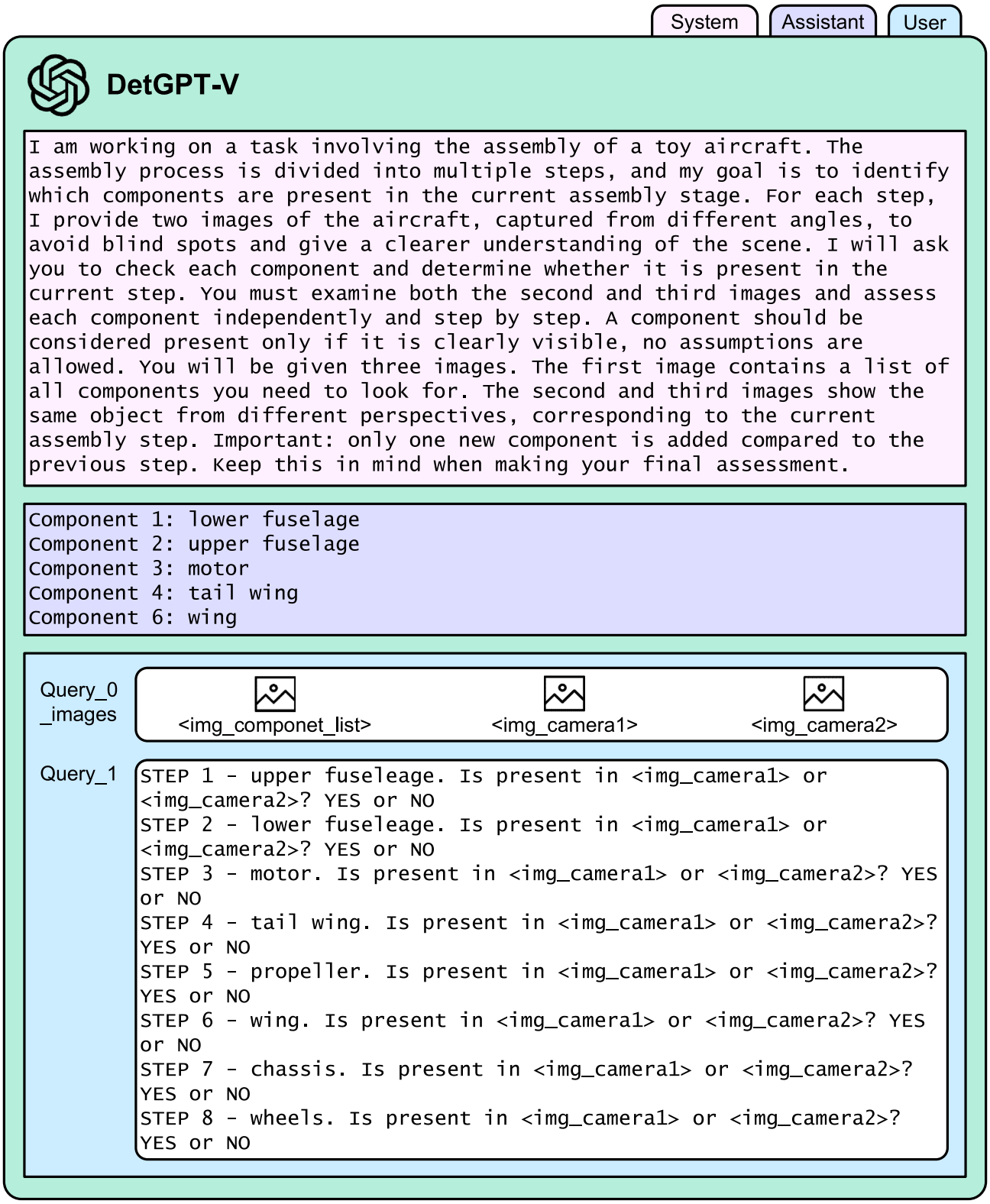

Fig. A.4. Prompt structure for the detection module DetGPT-V featuring three roles: system, assistant and user. The query is simplified answering the presence of each component with YES or NO.

References

[1] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, B. Ichter, D. Driess, J. Wu, C. Lu, M. Schwager, Foundation models in robotics: Applications, challenges, and the future, Int. J. Robot. Res. 44 (5) (2025) 701–739, http://dx.doi.org/10.1177/02783649241281508.

[2] D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y.L. Tan, L.Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, S. Levine, Octo: An open-source generalist robot policy, in: Robotics: Science and Systems, Deft, Netherlands, 2024.

[3] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, K. Ikeuchi, GPT-4v(ision) for robotics: Multimodal task planning from human demonstration, IEEE Robot. Autom. Lett. 9 (11) (2024) 10567–10574, http://dx.doi.org/10.1109/LRA.2024.3477090.

[4] K. Hori, K. Suzuki, T. Ogata, Enhancement of long-horizon task planning via active and passive modification in large language models, Sci. Rep. 15 (1) (2025) 7113.

[5] S. Li, P. Zheng, S. Liu, Z. Wang, X.V. Wang, L. Zheng, L. Wang, Proactive human–robot collaboration: Mutual-cognitive, predictable, and self-organising perspectives, Robot. Comput.-Integr. Manuf. 81 (2023) 102510, http://dx.doi.org/10.1016/j.rcim.2022.102510, URL https://www.sciencedirect.com/science/article/pii/S0736584522001922.

[6] H. Wang, K. Kedia, J. Ren, R. Abdullah, A. Bhardwaj, A. Chao, K.Y. Chen, N. Chin, P. Dan, X. Fan, G. Gonzalez-Pumariega, A. Kompella, M.A. Pace, Y. Sharma, X. Sun, N. Sunkara, S. Choudhury, MOSAIC: Modular foundation models for assistive and interactive cooking, in: P. Agrawal, O. Kroemer, W. Burgard (Eds.), Proceedings of the 8th Conference on Robot Learning, in: Proceedings of Machine Learning Research, vol. 270, PMLR, 2025, pp. 2220–2294, URL https://proceedings.mlr.press/v270/wang25h.html.

[7] L. Wang, R. Gao, J. Váncza, J. Krüger, X. Wang, S. Makris, G. Chryssolouris, Symbiotic human-robot collaborative assembly, CIRP Ann 68 (2) (2019) 701–726, http://dx.doi.org/10.1016/j.cirp.2019.05.002, URL https://www.sciencedirect.com/science/article/pii/S0007850619301593.

[8] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, ACM Comput. Surv. 55 (9) (2023) http://dx.doi.org/10.1145/3560815.

[9] W. Gurnee, M. Tegmark, Language models represent space and time, in: B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, Y. Sun (Eds.), in: International Conference on Learning Representations, vol. 2024, 2024, pp. 2483–2503, URL https://proceedings.iclr.cc/paper_files/paper/2024/file/0a6059857ae5c82ea9726ee9282a7145-Paper-Conference.pdf.

[10] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, A. Miller, Language models as knowledge bases? in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2463–2473, http://dx.doi.org/10.18653/v1/D19-1250, URL https://aclanthology.org/D19-1250/.

[11] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H.W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, J. Mach. Learn. Res. 24 (240) (2023) 1–113.

[12] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, et al., Simple open-vocabulary object detection, in: European Conference on Computer Vision, Springer, 2022, pp. 728–755.

[13] X. Xiao, J. Liu, Z. Wang, Y. Zhou, Y. Qi, S. Jiang, B. He, Q. Cheng, Robot learning in the era of foundation models: A survey, Neurocomputing 638 (2025) 129963.

[14] S. Zhao, S. Liu, Y. Jiang, B. Zhao, Y. Lv, J. Zhang, L. Wang, R.Y. Zhong, Industrial foundation models (IFMs) for intelligent manufacturing: A systematic review, J. Manuf. Syst. 82 (2025) 420–448, http://dx.doi.org/10.1016/j.jmsy.2025.06.011, URL https://www.sciencedirect.com/science/article/pii/S027861252500161X.

[15] J. Qi, L. Lu, F. Wang, H.-Y. Lee, D. Navarro-Alarcon, Z. Zhang, P. Zhou, LLM-driven symbolic planning and hierarchical imitation learning for long-horizon deformable object assembly, Robot. Comput.-Integr. Manuf. 97 (2026) 103096, http://dx.doi.org/10.1016/j.rcim.2025.103096, URL https://www.sciencedirect.com/science/article/pii/S0736584525001504.

[16] D. Kim, A. Angelova, W. Kuo, Region-aware pretraining for open-vocabulary object detection with vision transformers, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 11144–11154, http://dx.doi.org/10.1109/CVPR52729.2023.01072.

[17] A. Zareian, K.D. Rosa, D.H. Hu, S.-F. Chang, Open-vocabulary object detection using captions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14393–14402.

[18] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, et al., Grounded decoding: Guiding text generation with grounded models for embodied agents, Adv. Neural Inf. Process. Syst. 36 (2024).

[19] C. Zhang, J. Chen, J. Li, Y. Peng, Z. Mao, Large language models for human–robot interaction: A review, Biomim. Intell. Robot. 3 (4) (2023) 100131, http://dx.doi.org/10.1016/j.birob.2023.100131, URL https://www.sciencedirect.com/science/article/pii/S2667379723000451.

[20] H. Liu, C. Li, Q. Wu, Y.J. Lee, Visual instruction tuning, Adv. Neural Inf. Process. Syst. 36 (2024).

[21] J. Lim, S. Patel, A. Evans, J. Pimley, Y. Li, I. Kovalenko, Enhancing human-robot collaborative assembly in manufacturing systems using large language models, in: 2024 IEEE 20th International Conference on Automation Science and Engineering, CASE, 2024, pp. 2581–2587, http://dx.doi.org/10.1109/CASE59546.2024.10711843.

[22] Z. Liu, H. Sun, D. Yuan, Automatic analysis of alarm embedded with large language model in police robot, Biomim. Intell. Robot. 5 (3) (2025) 100220, http://dx.doi.org/10.1016/j.birob.2025.100220, URL https://www.sciencedirect.com/science/article/pii/S2667379725000117.

[23] S. Sun, C. Li, Z. Zhao, H. Huang, W. Xu, Leveraging large language models for comprehensive locomotion control in humanoid robots design, Biomim. Intell. Robot. 4 (4) (2024) 100187, http://dx.doi.org/10.1016/j.birob.2024.100187, URL https://www.sciencedirect.com/science/article/pii/S2667379724000457.

[24] D. Han, T. McInroe, A. Jelley, S.V. Albrecht, P. Bell, A. Storkey, LLM-personalize: Aligning LLM planners with human preferences via reinforced self-training for housekeeping robots, in: O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B.D. Eugenio, S. Schockaert (Eds.), Proceedings of the 31st International Conference on Computational Linguistics, Association for Computational Linguistics, Abu Dhabi, UAE, 2025, pp. 1465–1474, URL https://aclanthology.org/2025.coling-main.98/.

[25] Y. Wang, Q. Guo, L. Zheng, B. Wang, P. Zheng, Z. Qi, LLM based autonomous agent of human-robot collaboration for aerospace wire harnessing assembly, Robot. Comput.-Integr. Manuf. 97 (2026) 103120, http://dx.doi.org/10.1016/j.rcim.2025.103120, URL https://www.sciencedirect.com/science/article/pii/S0736584525001747.

[26] P. Ding, J. Zhang, P. Zhang, H. Li, D. Wang, CCM-FCC: LLM-powered cognition-centered AI agent framework for proactive human-robot collaboration, Robot. Comput.-Integr. Manuf. 98 (2026) 103145, http://dx.doi.org/10.1016/j.rcim.2025.103145, URL https://www.sciencedirect.com/science/article/pii/S0736584525001991.

[27] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), in: Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[28] H.M. Ahmad, A. Rahimi, Deep learning methods for object detection in smart manufacturing: A survey, J. Manuf. Syst. 64 (2022) 181–196.

[29] Z.-Q. Zhao, P. Zheng, S.-t. Xu, X. Wu, Object detection with deep learning: A review, IEEE Trans. Neural Netw. Learn. Syst. 30 (11) (2019) 3212–3232.

[30] K. Geng, J. Qiao, N. Liu, Z. Yang, R. Zhang, H. Li, Research on real-time detection of stacked objects based on deep learning, J. Intell. Robot. Syst. 109 (4) (2023) 82.

[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2021.

[32] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.-Y. Lo, P. Dollar, R. Girshick, Segment anything, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2023, pp. 4015–4026.

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

[34] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, T. Duerig, Scaling up visual and vision-language representation learning with noisy text supervision, in: International Conference on Machine Learning, PMLR, 2021, pp. 4904–4916.

[35] N.M.M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, A. Szlam, CLIP-fields: Weakly supervised semantic fields for robotic memory, 2022, CoRR.

[36] Z. Wang, J. Yu, A.W. Yu, Z. Dai, Y. Tsvetkov, Y. Cao, SimVLM: Simple visual language model pretraining with weak supervision, in: International Conference on Learning Representations, 2022.

[37] X. Gu, T.-Y. Lin, W. Kuo, Y. Cui, Open-vocabulary object detection via vision and language knowledge distillation, in: International Conference on Learning Representations, 2022.

[38] L.H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, et al., Grounded language-image pre-training, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10965–10975.
[39] M. Liu, Y. Zhu, H. Cai, S. Han, Z. Ling, F. Porikli, H. Su, Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 21736–21746.
[40] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, N. Carion, Mdetr-modulated detection for end-to-end multi-modal understanding, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1780–1790.
[41] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M.G. Arenas, H.-T.L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al., Language to rewards for robotic skill synthesis, in: 7th Annual Conference on Robot Learning, 2023.
[42] A. Zeng, M. Attarian, K.M. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M.S. Ryoo, V. Sindhwani, J. Lee, et al., Socratic models: Composing zero-shot multimodal reasoning with language, in: The Eleventh International Conference on Learning Representations, 2022.
[43] W. Huang, P. Abbeel, D. Pathak, I. Mordatch, Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, in: International Conference on Machine Learning, PMLR, 2022, pp. 9118–9147.
[44] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R.M.J. Ruano, K. Jeffrey, S. Jesmonth, N.J. Joshi, R.C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D.M. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, Do as I can, not as I say: Grounding language in robotic affordances, in: Conference on Robot Learning, 2022, URL https://api.semanticscholar.org/CorpusID:247939706.
[45] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al., Inner monologue: Embodied reasoning through planning with language models, in: Conference on Robot Learning, PMLR, 2023, pp. 1769–1782.
[46] B. Zhang, H. Soh, Large language models as zero-shot human models for human-robot interaction, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 7961–7968.
[47] H. Liu, Y. Zhu, K. Kato, A. Tsukahara, I. Kondo, T. Aoyama, Y. Hasegawa, Enhancing the LLM-based robot manipulation through human-robot collaboration, IEEE Robot. Autom. Lett. 9 (8) (2024) 6904–6911, http://dx.doi.org/10.1109/LRA.2024.3415931.
[48] H. Huang, F. Lin, Y. Hu, S. Wang, Y. Gao, Copa: General robotic manipulation through spatial constraints of parts with foundation models, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2024, pp. 9488–9495.
[49] K. Fang, F. Liu, P. Abbeel, S. Levine, Moka: Open-world robotic manipulation through mark-based visual prompting, Robot.: Sci. Syst. (2024).
[50] Y. Ding, X. Zhang, C. Paxton, S. Zhang, Task and motion planning with large language models for object rearrangement, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 2086–2092.
[51] Z. Xu, H.-T.L. Chiang, Z. Fu, M.G. Jacob, T. Zhang, T.-W.E. Lee, W. Yu, C. Schenck, D. Rendleman, D. Shah, F. Xia, J. Hsu, J. Hoech, P. Florence, S. Kirmani, S. Singh, V. Sindhwani, C. Parada, C. Finn, P. Xu, S. Levine, J. Tan, Mobility VLA: Multimodal instruction navigation with long-context vlms and topological graphs, in: P. Agrawal, O. Kroemer, W. Burgard (Eds.), Proceedings of the 8th Conference on Robot Learning, in: Proceedings of Machine Learning Research, vol. 270, PMLR, 2025, pp. 3866–3887, URL https://proceedings.mlr.press/v270/xu25b.html.
[52] C. Huang, O. Mees, A. Zeng, W. Burgard, Visual language maps for robot navigation, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 10608–10615.
[53] X. Zhao, M. Li, C. Weber, M.B. Hafez, S. Wermter, Chat with the environment: Interactive multimodal perception using large language models, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 3590–3596.
[54] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, L. Fei-Fei, VoxPoser: Composable 3D value maps for robotic manipulation with language models, in: Conference on Robot Learning, PMLR, 2023, pp. 540–562.
[55] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A.P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, et al., Scaling vision transformers to 22 billion parameters, in: International Conference on Machine Learning, PMLR, 2023, pp. 7480–7512.
[56] Y. Hu, F. Lin, T. Zhang, L. Yi, Y. Gao, Look before you leap: Unveiling the power of GPT-4V in robotic vision-language planning, in: First Workshop on Vision-Language Models for Navigation and Manipulation At ICRA 2024.
[57] Y. Jin, D. Li, A. Yong, J. Shi, P. Hao, F. Sun, J. Zhang, B. Fang, Robotgpt: Robot manipulation learning from chatgpt, IEEE Robot. Autom. Lett. (2024).
[58] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, A.M. Dollar, Benchmarking in manipulation research: Using the yale-CMU-berkeley object and model set, IEEE Robot. Autom. Mag. 22 (3) (2015) 36–52, http://dx.doi.org/10.1109/MRA.2015.2448951.
[59] T. Silver, S. Dan, K. Srinivas, J.B. Tenenbaum, L. Kaelbling, M. Katz, Generalized planning in pddl domains with pretrained large language models, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (18) 2024, pp. 20256–20264.
[60] A. Curtis, N. Kumar, J. Cao, T. Lozano-Pérez, L.P. Kaelbling, Trust the PRoC3S: Solving long-horizon robotics problems with LLMs and constraint satisfaction, in: Conference on Robot Learning, PMLR, 2025, pp. 1362–1383.
[61] K. Obata, T. Aoki, T. Horii, T. Taniguchi, T. Nagai, Lip-LLM: Integrating linear programming and dependency graph with large language models for multi-robot task planning, IEEE Robot. Autom. Lett. 10 (2) (2025) 1122–1129, http://dx.doi.org/10.1109/LRA.2024.3518105.
[62] 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, Adv. Neural Inf. Process. Syst. 35 (2022) 24824–24837.
[63] J. Materzynska, G. Berger, I. Bax, R. Memisevic, The jester dataset: A large-scale video dataset of human gestures, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019.
[64] Y. Chen, W. Cui, Y. Chen, M. Tan, X. Zhang, J. Liu, H. Li, D. Zhao, H. Wang, RoboGPT: An LLM-based long-term decision-making embodied agent for instruction following tasks, IEEE Trans. Cogn. Dev. Syst. 17 (5) (2025) 1163–1174, http://dx.doi.org/10.1109/TCDS.2025.3543364.