



OPEN ACCESS

EDITED BY

Ata Jahangir Moshayedi,
Jiangxi University of Science and
Technology, China

REVIEWED BY

Marco Roveri,
University of Trento, Italy
Zhongpan Zhu,
University of Shanghai for Science and
Technology, China

*CORRESPONDENCE

Alireza Rastegarpanah,
✉ a.rastegarpanah@aston.ac.uk
Cansu Erdogan,
✉ cxa215@student.bham.ac.uk

†These authors share first authorship

RECEIVED 17 October 2025

REVISED 21 December 2025

ACCEPTED 19 January 2026

PUBLISHED 27 March 2026

CITATION

Erdogan C, Contreras CA,
Rastegarpanah A, Chiou M and Stolkin R
(2026) Intent-driven LLM ensemble
planning for flexible multi-robot
manipulation.
Front. Robot. AI 13:1727433.
doi: 10.3389/frobt.2026.1727433

COPYRIGHT

© 2026 Erdogan, Contreras,
Rastegarpanah, Chiou and Stolkin. This
is an open-access article distributed
under the terms of the [Creative
Commons Attribution License \(CC BY\)](#).
The use, distribution or reproduction in
other forums is permitted, provided the
original author(s) and the copyright
owner(s) are credited and that the
original publication in this journal is
cited, in accordance with accepted
academic practice. No use, distribution
or reproduction is permitted which
does not comply with these terms.

Intent-driven LLM ensemble planning for flexible multi-robot manipulation

Cansu Erdogan ^{1*†}, Cesar Alan Contreras ^{1†},
Alireza Rastegarpanah ^{1,2*†}, Manolis Chiou ³ and
Rustam Stolkin¹

¹Extreme Robotics Lab, School of Metallurgy and Materials, University of Birmingham, Birmingham, United Kingdom, ²Department of Applied Artificial Intelligence and Robotics, School of Computer Science, Aston University, Birmingham, United Kingdom, ³School of Electronic Engineering and Computer Science, Queen Mary University of London, London, United Kingdom

Introduction: In this study, we address intent-driven task planning for complex multi-action manipulation sequences in heterogeneous multi-robot cells. Given a perception back-end that outputs a structured object-level scene description and a human operator's natural-language intent, we generate a precedence-consistent object-level robot-action sequence, which can then be executed by passing each such action to a lower-level motion planning module.

Methods: The pipeline integrates i. perception-to-text scene encoding, ii. an ensemble of large language models (LLMs) that generate candidate action sequences based on the operator's intent, iii. an LLM-based verifier that enforces formatting and precedence constraints, and iv. a deterministic consistency filter that rejects hallucinated objects. The pipeline is evaluated on an example task in which two robot arms work collaboratively to dismantle an electric-vehicle (EV) battery for recycling applications. A variety of components must be grasped and removed in specific sequences, determined either by human instructions or by task-order feasibility decisions made by the autonomous system.

Results: On 200 real scenes with 600 operator prompts across five component classes, we used metrics of full-sequence correctness and next-task correctness to evaluate and compare five LLM-based planners (including ablation analyses of pipeline components). We also evaluated the LLM-based human interface in terms of time to execution and NASA TLX using human participant experiments. On 200 real scenes and 600 prompts, full-sequence correctness improves from 0.761 (single LLM) to 0.824 (6-LLM + verifier + deterministic filter), and next-object correctness improves from 0.866 to 0.894.

Discussion: Results in our case study indicate that our ensemble-with-verification approach reliably maps operator intent to safe multi-robot plans while maintaining low user effort.

KEYWORDS

human-robot interaction, intent recognition, large language models, multi-robot disassembly, task planning

1 Introduction

In this study, we address the problem of planning complex manipulation tasks, in which multiple robots with different end-effectors and capabilities must plan and execute concatenated sequences of actions in unstructured scenes, informed

by i. perception-to-text scene encoding and ii. human operator intent, inferred from simple natural language instructions.

Traditional, pre-programmed automation has been effective in highly structured settings, gaining widespread adoption in automotive manufacturing and other factory assembly tasks since the 1970s. However, unstructured, variable, and uncertain environments pose additional challenges, which are a bottleneck to the robotization of many other societally important industries. Robotic systems now span diverse application domains, from micro-robotics in healthcare to service robots and mobile AGVs and to hazardous environments such as end-of-life disassembly and nuclear decommissioning (Moshayedi et al., 2024a; b, 2025; Erdogan et al., 2024; Contreras et al., 2025b; Vitanov et al., 2021; Bird et al., 2021; Talha et al., 2016; Marturi et al., 2016; Marturi et al., 2017; Stolkin et al., 2023; Agency, 2025).

The disassembly of electric-vehicle (EV) batteries is a prototypical task that demands strong generalization from multi-robot systems. Battery packs vary widely across makes, models, and model years; there is no stable standard, and the designs evolve continuously, so the classic “re-program on every new variant” approach is untenable (Stolkin et al., 2024). Instead, scalable automation must i. use computer vision to robustly recognize and localize heterogeneous components and infer how they fit together; ii. plan low-level actions (grasping, moving, unscrewing, and cutting) for specific robots on specific parts under perception uncertainty; iii. schedule and coordinate multi-robot, multi-action task sequences; and iv. provide an intuitive, high-level interface so that a recycling worker (without robotics expertise) can quickly instruct the system. Similar needs arise in other unstructured settings (e.g., sorting and separating mixed waste streams for recycling and remote decommissioning of contaminated legacy facilities in high-hazard environments). Traditional task and motion planning (TAMP) pipelines, which rely on handcrafted rules and rigid schedules, struggle to handle such variability (Asif et al., 2024), motivating the intent-driven, perception-grounded approach we pursue in this study.

Recent advances in large-scale pre-trained language models (LLMs) offer new opportunities for flexible, high-level reasoning over symbolic and spatial descriptions. By leveraging LLMs’ ability to process and synthesize contextual information, it becomes feasible to generate executable multi-robot plans from textual scene descriptions combined with operator intent. Coupled with perception for perceptual grounding of task representations and motion planning for execution, this capability reduces manual modeling effort and enables a more adaptive, scalable planning pipeline.

1.1 Overview

In this study, we introduce a multi-LLM-based task planning framework for multi-robot disassembly operations. A user writes what they want to be taken apart, and the system turns that request into a step-by-step plan that two robots can safely carry out. More specifically, the approach integrates vision-based scene understanding modules with an LLM-driven reasoning layer to infer object ordering and produce abstract task plans. These are then grounded into executable sequences via a motion-level planner, which enforces physical constraints and robot coordination. We

conduct a structured evaluation based on a custom dataset of battery disassembly scenarios, using metrics such as task ordering accuracy and plan execution success rates. The final pipeline is shown in Figure 1.

This work makes four key contributions: i. an end-to-end, language-conditioned disassembly pipeline, which maps operator intent and live perception to executable, precedence-constrained action lists for multi-arm systems (demonstrated on a two-arm cell); ii. a domain-agnostic perception–language interface, which serializes detections from text-output vision models (YOLOv8) as structured text with coordinates in the operator frame, enabling spatial reasoning while decoupling detection from planning; iii. a planner, which uses a single instruction-tuned checkpoint (Qwen3-32B) with stochastic sampling to produce diverse candidates, followed by an LLM verifier and a deterministic consistency filter to enforce precedence and format, and reject objects that are ungrounded; and iv. a workload-aware human-in-the-loop execution layer, which coordinates multiple manipulators, using a digital twin for collision checks and synchronized trajectory streaming (MoveIt), providing a simplified hand-off from plan to execution to reduce operator cognitive and interaction workload.

1.1.1 Scope and system boundaries

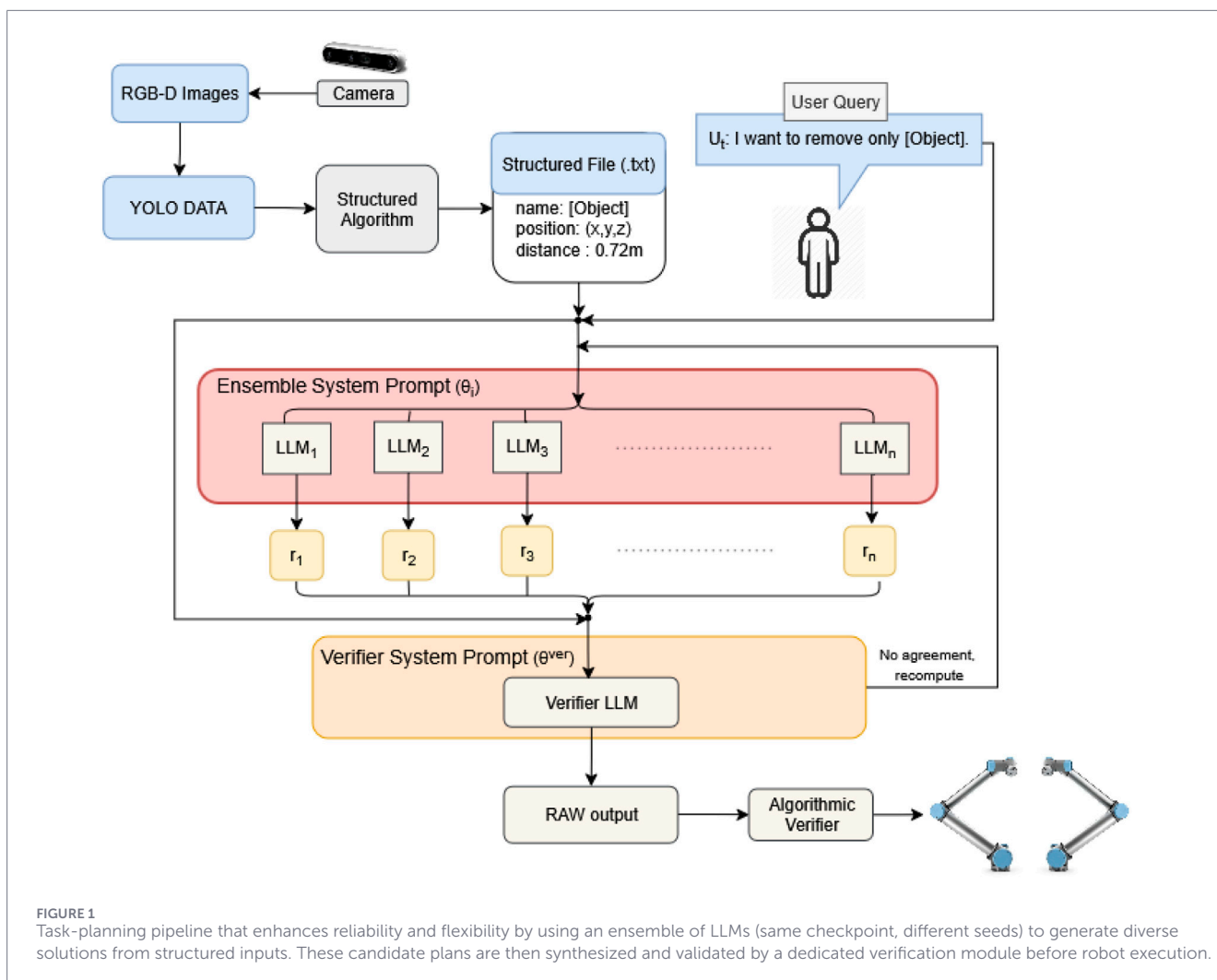
This study contributes to the intent-to-plan layer, mapping a structured scene state and operator intent to a precedence-consistent, object-level disassembly sequence, with explicit verification and deterministic grounding before execution. In our framework, the perception (YOLOv8 in our prototype) and motion planning/execution (MoveIt/OMPL in our prototype) steps are interchangeable backends and are not the contribution of this work. For the research, we evaluate three questions: (Q1) Does ensembling + verification improve plan correctness relative to a single LLM baseline? (Q2) What is the effect of the deterministic consistency filter on grounding and plan quality? and (Q3) Can non-expert users command the two-arm cell using short language intents with low workload and reduced time-to-action?

2 Literature review

Our setup comprises multiple components, including task planning, large language models, ensemble methods, operator intent, and adaptive autonomy. In this section, we review the relevant work for each component and motivate the design choices used later.

2.1 Task planning and robot allocation

In multi-robot systems, task planning involves selecting and sequencing a set of high-level actions that collectively achieve a system-level objective. At the same time, robot allocation assigns these actions to individual agents based on capabilities, resources, availability, and spatial constraints. Coordination between both layers is necessary to ensure efficient and collision-free movements in shared environments. Classical optimization methods are often used, such as mixed-integer linear programming (MILP) and constraint programming (CP), which encode task



assignment under resource and precedence constraints (Fatemi-Anaraki et al., 2023; Touzani et al., 2022). These approaches are very effective in structured industrial scenarios. Still, they may suffer from scalability issues and experience degraded performance in dynamic or uncertain environments.

To improve scalability and adaptation, recent works have explored the use of heuristic and distributed scheduling methods (Wang and Chen, 2024; Pan et al., 2024), which show more responsive and decentralized decision-making capabilities under uncertainties. Knowledge-based and semantic reasoning frameworks (Bernardo et al., 2023) have introduced domain semantics awareness into task planning to interpret object roles, workspace conditions, and task dependencies. Despite progress, many frameworks still rely on handcrafted rules, limiting their adaptability to new tasks or environments. These limits motivate data-driven approaches that can adapt with less manual modeling.

We replace much of the handcrafted task-ordering logic with an LLM ensemble that infers precedence-consistent removal sequences from general disassembly instructions, a text-based scene encoding, and an operator prompt. We then double-check these sequences with an LLM verifier and a deterministic data-consistency filter

before any motion is executed. The framework also lets the operator edit the plans and take control at any time.

2.2 Large language models on robotics

The integration of LLMs into robotics has opened avenues for high-level task planning, semantic perception, and human–robot interaction (HRI). Among early impactful frameworks, RoboLLM demonstrated strong performance on vision-language benchmarks such as ARMBench, showing how LLMs can be grounded in visual inputs for robotic manipulation tasks (Long et al., 2024). However, these benchmarks typically represent static or highly structured environments. Real-world multi-robot systems tend to operate under unstructured and uncertain conditions.

Working under these constraints, the 3D-LOTUS++ framework in Towards Generalizable Vision-Language Robotic Manipulation combines the high-level reasoning of LLMs with the spatial awareness of vision language models (VLMs) for object localization. This integration improves generalization in robotic manipulation (Garcia et al., 2025). Another framework working in this direction is ManipLLM, which introduces a multimodal LLM-based manipulation system that focuses on object-centric

reasoning (Li et al., 2024). Unlike earlier learning-based methods limited to predefined object categories, ManipLLM uses commonsense reasoning chains and test-time adaptation strategies to generalize across diverse manipulation tasks, performing well in simulated and real-world environments.

For HRI, recent work has emphasized collaboration with LLMs, including studies on LLM-enabled robots performing execution, negotiation, selection, and plan generation (Kim et al., 2024), along with vision–language–action reasoning combined with teleoperation (Liu et al., 2024). Findings suggest that although LLMs excel in affective and negotiation-based interactions, they struggle to maintain logical consistency and support creative collaboration.

End-to-end VLMs and vision–language–action (VLA) policies can map images and language directly to robot actions. These approaches can generalize broadly, but they are difficult to inspect and validate in safety-critical industrial settings, and they typically require domain-aligned robot data and validation in the target cell. For disassembly, we instead use a modular perception–plan–motion architecture with an explicit, human-readable intermediate plan and deterministic grounding (Driess et al., 2023; Brohan et al., 2023). In our work, we operationalize LLM planning for disassembly tasks by i. converting RGB-D detections into a structured text scene, ii. sampling multiple plans from a single LLM checkpoint with different seeds, iii. verifying logical ordering and precedence with an LLM instructed for verification tasks, and iv. applying a final algorithmic data-consistency filter to reduce hallucinations before commanding the robots.

2.3 Ensembles and instruction tuning

Instruction tuning (IT) adapts a pretrained language model to follow natural-language prompts by fine-tuning it on instruction triples (instruction–input–output). The supervised fine-tuning (SFT) pipeline begins with a training stage over dozens to hundreds of different tasks (Wei et al., 2022) and sometimes is followed by a reinforcement learning step with human feedback (RLHF) using their preference data (Ouyang et al., 2022). Exposing the systems to highly diverse instructional data and human feedback provides strong zero-shot generalization on unseen tasks and also helps reduce toxic behaviors (Wei et al., 2022; Ouyang et al., 2022; Chung et al., 2022). Early evidence of the strength of SFT came from FLAN, which fine-tuned a 137B parameter model on 60+ different tasks and surpassed bigger models in zero-shot accuracy (Wei et al., 2022), with subsequent works showing that scaling the number of tasks and the model parameters improves instruction-following abilities and that injecting chain-of-thought traces during fine-tuning also boosts reasoning accuracy (Chung et al., 2022).

The increase in IT models prompted the design of specialized benchmarks that evaluate models on their obedience to explicit constraints, some of which include IFEval, which judges whether generated text satisfies verifiable instructions (e.g., length or lexical constraints) (Zhou et al., 2023); FollowBench, which dissects adherence across content, format, and style dimensions (Jiang et al., 2024); and M-IFEval, which adds judgment of multilingual instructions, comparing it to the default IFEval (Dussolle et al., 2025).

Ensemble methods can complement instruction tuning. Deep ensembles improve calibration by aggregating independently sampled models (Lakshminarayanan et al., 2017). Model-library selection ensembles pick a diverse subset from thousands of candidates for optimal accuracy (Caruana et al., 2004). Sparse mixture-of-expert (MoE) architectures, using the Switch Transformer, scale to trillions of parameters, and some obtain ensemble-like gains by fusing final layers at fixed compute by routing a subset of tokens through selected layers (Fedus et al., 2021; Raposo et al., 2024). Ensembles of instruction-tuned generators, in theory, could produce higher-quality synthetic training data and enable self-consistency voting across multiple IT models, reducing reasoning errors at inference. Yet, despite the progress in each direction, studies on ensemble design choices tailored to instruction-following constraints remain limited.

In our study, instead of extra fine-tuning, we perform an ensemble-at-inference on a single instruction-tuned LLM checkpoint (in a 1/3/6 ensemble of Qwen3-32B) and add an LLM verifier for format, precedence, and logical planning verification. We also enforce an algorithmic filter that checks for hallucinations as this process is faster and more adaptable than retraining.

2.4 Operator intent

The ability to understand and respond to an operator's intentions is a prerequisite for fluent human–robot teaming. Early Bayesian formulations from motion studies showed that predicting a collaborator's next goal allows a robot to adapt its plan, yielding improvements in both objective and perceived performance (Liu et al., 2016). Some works have also framed intent recognition as an online classification problem; in teleoperation work, for example, navigation and manipulation intents are able to be inferred and classified with onboard sensing, robot trajectories, saliency, and geometric cues (Tsagkournis et al., 2023; Contreras et al., 2025a; Panagopoulos et al., 2021) reaching real-time accuracy compared to hand-engineered baselines.

Vision-only, zero-shot intent recognition has also shown reductions in operator effort on unseen manipulation tasks (Belsare et al., 2025). In multi-robot settings, inferring collective operator intent introduces additional operator and bandwidth challenges, with some researchers on human–swarm interaction marking cognitive load as the main bottleneck when conveying intent to dozens of agents simultaneously (Kolling et al., 2016). Shared-control systems take it one step further by blending user commands with autonomy in proportion to the certainty of inferred intent, and sometimes, intent-aware assistance is preferred over full autonomy (Bowman et al., 2024).

Taking from the literature the importance of assistance and intent recognition, in this study, we treat natural-language prompts as the explicit operator intent and fuse them with the scene text so the ensemble orders only disassembly steps that satisfy the prompt (e.g., “remove only leafcell”), while the verifier enforces logical disassembly precedence and the filter prevents objects not present in the scene from appearing. Providing these calculations to the system keeps cognitive-load low, while it also preserves trust by allowing operators to verify and change the disassembly order before execution.

2.5 Variable autonomy

Variable autonomy (VA) refers to a robot's ability to modulate its level of independence, shifting decisions and control between human operators and onboard autonomy in response to context, workload, mission phase, or robot conditions (Contreras et al., 2025b; Methnani et al., 2024; Chiou et al., 2021; Reinmund et al., 2024). Some adjustable-autonomy frameworks formalize transfer of control strategies, balancing decision quality against miscoordination costs, and show performance gains for single operators supervising several robots at varying autonomy levels (Scerri et al., 2002; Crandall and Goodrich, 2001). More recently, some research has framed VA as a prerequisite for trustworthy AI, emphasizing transparency and explainability as determinants of when autonomy should be ceded or reclaimed (Methnani et al., 2024).

Language-centric VA mechanisms in multi-robot tasks on VR testbeds have also been explored (Lakhnati et al., 2024) and have been shown to be helpful in deciding when to act autonomously and when to ask an operator for help, further expanding the available methods for implementing variable autonomy. Variable autonomy rules and allocation rules can also be learned; for example, the ATA-HRL framework utilizes hierarchical reinforcement learning to re-allocate tasks and determine autonomy mode in multi-human–multi-robot teams, outperforming fixed-autonomy baselines under dynamic mission conditions (Yuan et al., 2025).

Our planner supports on-the-fly autonomy adjustment via language; operators may change objectives mid-execution, while the LLM verifier and algorithmic data-consistency filter preserve safety. A motion layer allows the coordination of multiple arms without restarting, enabling the system to maintain a low cognitive workload, and it also lets operators take control at any time.

3 Methods

When planning for robot disassembly, task hierarchies tend to be static. However, this limitation prevents systems from adapting to environmental changes. With vision models, it has been possible to adapt to changing positions; however, adapting to different operator intentions and explicit needs remains a challenge. To evaluate our dynamically adaptive method, we build on our previous studies (Erdogan et al., 2024; Shaarawy et al., 2025) and design a robotic scene consisting of two UR10e robots with cameras mounted on their end effectors, tasked with identifying objects that form part of a disassembly task (Figure 2) and correctly disassembling them. For this task, we used the weights of YOLOv8 trained on 226 images from our previous study (Shaarawy et al., 2025) in a real-world environment, in addition to the use of the pre-trained Qwen-32B LLM, without any additional fine-tuning.

We deployed our experimental setup across two dedicated desktop computers. The first system, denoted as PC₁, was responsible for real-time perception and robotic control. PC₁ featured an Intel i7 CPU, an NVIDIA GTX 1080 Ti GPU, and 32 GB of RAM. This machine was directly connected to the Intel RealSense 435i RGB-D cameras mounted on the robots (Figure 3), which continuously provided synchronized RGB and depth data. Additionally, PC₁ managed low-level control of two universal robot (UR-series) manipulators, handling joint commands and execution of planned motions.



FIGURE 2 Example perception output used by the LLM-ensemble planner. A YOLOv8 model overlays class labels and confidence scores for the battery components (Leafcell, Cable, Busbar, Service Plug, and Screw).

The second system, designated PC₂, performed high-level reasoning tasks and hosted the language models utilized in our experiments. PC₂ was configured with an Intel i9 CPU, an NVIDIA RTX 5090 GPU, and 32 GB of RAM. This computer exclusively ran a quantized (4-bit) version of the Qwen-32B large language model. All models were served through an Ollama-compatible API, and inter-system communication between PC₁ and PC₂ occurred over a WiFi network, with 200 ms latency.

3.1 Scene encoding

Raw sensor state $x_t \in \mathbb{R}^n$ is segmented by YOLOv8, yielding

$$s_t = g(x_t) = \{(b_k, c_k)\}_{k=1}^{K_t}.$$

The set s_t is rendered as an unstructured string and then structured and serialized, $\sigma_t = \text{Serialise}(s_t)$.

Excerpt of unstructured text s_t	Excerpt of structured text σ_t
THIS IS THE PUBLISHED DATA (dist mode: XYZ) LeafCell pos [1.304 0.877–0.081] dist = 1.574 bbox(m): TL [1.454 0.873 0.048] TR [1.455 0.884–0.202] BR [1.143 0.882–0.204] BL [1.141 0.870 0.046] C [0.189 0.003 0.669] LeafCell pos [1.309 0.854 0.186] dist = 1.574 ... Cable pos [1.516 0.900 0.059] dist = 1.764 ... Screw pos [1.167 0.852 0.092] dist = 1.448 ... Screw pos [1.438 0.846 0.270] dist = 1.690 ... Screw pos [1.431 0.854 0.103] dist = 1.669 ... Screw pos [1.169 0.866 0.032] dist = 1.455 ... Screw pos [1.167 0.843 0.285] dist = 1.467 ...	Group 1: LeafCell is in location [1.304, 0.877, –0.081] with screw [1.169, 0.866, 0.032] on top of it Group 2: LeafCell is in location [1.309, 0.854, 0.186] with screw [1.167, 0.852, 0.092], screw [1.438, 0.846, 0.27], screw [1.431, 0.854, 0.103], screw [1.167, 0.843, 0.285] on top of it Group 3: Cable is in location [1.516, 0.9, 0.059]

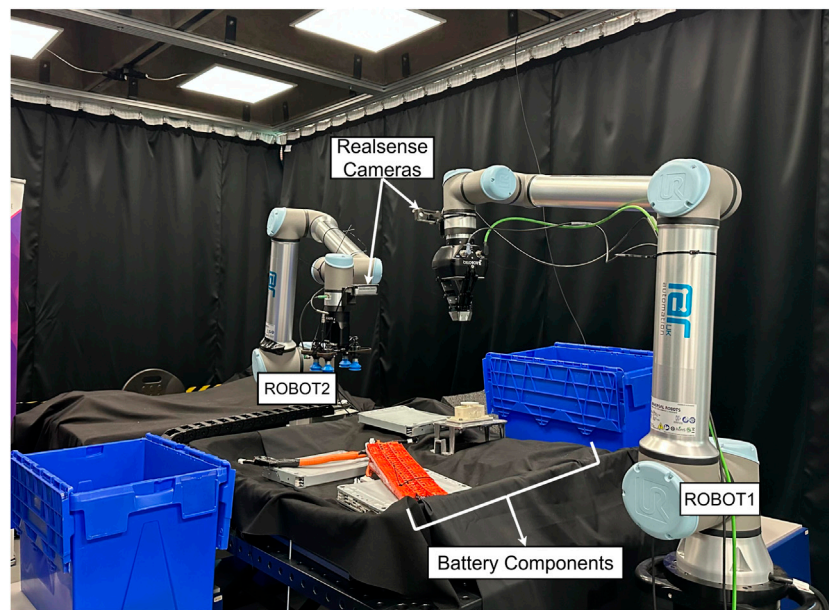


FIGURE 3
Data-collection setup: two UR10e collaborative robot arms (ROBOT1 and ROBOT2) equipped with Intel RealSense RGB-D cameras and the components arranged on the workbench.

3.2 Coordinate rewrite

Every Cartesian triple $[x, y, z]$ inside σ_t is replaced by $[-z, x, y, d]$ using Equation 1:

$$d = \sqrt{x^2 + y^2 + z^2}, \quad (1)$$

due to the transformations from the camera to spatial positions on the operator frame, while all other tokens are left untouched. The transformed scene text produced by the coordinate rewriting is provided in Equation 2:

$$D_t = \mathcal{T}(\sigma_t). \quad (2)$$

3.3 Generator-system prompt

The prompt θ_i transforms the backbone model into a *Generator*: from the structured scene description and the operator's instruction, it must produce, without commentary, an ordered list of objects that must be removed. To keep the generator and verifier perfectly aligned, the prompt first clarifies the role and output contract and then reminds the model of the coordinate convention already applied to the received input. A compact input-grammar description follows, covering how leads, followers, and their positions appear in each group statement. The prompt then details a domain-agnostic target-resolution algorithm that defines how to locate requested objects, accumulate any mandatory blockers, and order the final removal set. A series of self-checks ensures that the list is complete, free of duplicates, and respects all ordering rules before emission. Explicit error strings allow the downstream runner to detect fatal conditions early, while a disallowed-action section forbids headings, blank lines, fabricated data, or any explanatory text. Finally, a bank

of illustrative examples anchors the model's behavior on different corner cases without revealing internal task labels.

Generator system prompt θ_i

- Role declaration
- Coordinate convention
- Input grammar description
- Target resolution algorithm
- Self-checklist
- Error strings
- Disallowed actions
- Illustrative examples

3.4 Generator pass

Let $u_t^{\text{extra}} \in \mathcal{U}$ be the operator instruction and $\theta_i \in \Theta$ be the static system prompt for model i . The user-role message is U_t .

Final user message U_t

Here are the data to use: $\langle D_t \rangle$
This is what I want to do 'extra prompt': $\langle u_t^{\text{extra}} \rangle$
/no_think

With sampling seed s_j , the message sent to the LLM API is shown in Equation 3:

$$p_{i,j,t} = [(\text{"system"}, \theta_i), (\text{"user"}, U_t)]. \quad (3)$$

Generator LLM message

"System," θ_i
"User," U_t

3.5 LLM candidate generation

Let S be the (finite) set of sampling seeds for the single backbone model φ_i . For every $s_j \in S$, the model processes the payload $p_{i,j,t}$ from Equation 3 and returns a chat trace, as shown in Equation 4:

$$c_{i,j,t} = \varphi_i(p_{i,j,t}). \tag{4}$$

Define the function `last:trace` \rightarrow string that keeps only the final bullet list of a trace. The seed-specific candidate text is therefore

$$r_{i,j,t} = \text{last}(c_{i,j,t}). \tag{5}$$

3.6 Verifier system prompt logic

The verifier system prompt θ^{ver} guides the LLM into a verification-focused role, independent from the previous generation step. Its purpose is to enforce consistency and correctness among candidate responses. The prompt is explicitly structured into functional sections, starting with a clear role definition, followed by a description of the structured input layout provided by the user: this includes the initial context (system and user prompts) and candidate responses produced earlier.

A concise summary of the reference algorithm then follows, outlining how ground truth should be recomputed from the structured input without relying on task-specific terminology. Subsequently, the prompt defines multiple categories of validation criteria, including strict formatting requirements to ensure syntactic uniformity, presence constraints that mandate exact matches within the context, ordering rules to maintain algorithmic consistency, and a policy to eliminate duplicate entries. After validation, the verifier is tasked with employing a simple decision procedure, selecting the first candidate list that fulfills all specified criteria. If no candidate meets all conditions, the verifier must explicitly return an empty output, prohibiting any form of merging, partial acceptance, or speculative editing.

Finally, the prompt concludes by precisely specifying the required output format and explicitly restricting actions that fall outside its scope, such as commentary, partial lists, or explanations of additional context. This design systematically ensures deterministic and reliable verification without disclosing or embedding unnecessary domain knowledge, enabling it to function with different unspecified object labels.

```

Verifier system prompt  $\theta^{\text{ver}}$ 
- Role and responsibilities definition
- Structured input description
- Generalized algorithm recap
- Validation criteria
- Decision and selection logic
- Explicit output formatting rules
- Restricted actions and prohibitions
    
```

3.7 Verifier pass

Let

$$C_t = \text{str}(\theta_i, U_t)$$

be the printable pair comprising the generation system prompt θ_i and user message U_t , as denoted by $\mathcal{R}_t = \{r_{j,t} \mid s_j \in S\}$. The set of seed-specific candidate lists is generated in Equation 5. A fixed deterministic formatter $\Phi: (\text{context}, \text{set of strings}) \rightarrow \text{string}$ produces the user-side input to the verifier:

$$\xi_t = \Phi(C_t, \mathcal{R}_t).$$

The same LLM checkpoint is invoked once more, now under a dedicated verifier system prompt θ^{ver} ; its output is

$$v_t = g(\theta^{\text{ver}}, \xi_t),$$

A single cleaned bullet list in plain text is as follows:

```

Verifier LLM message
System:  $\theta^{\text{ver}}$ 
User:
= ORIGINAL CONTEXT ==
<C_t>
=== CANDIDATE RESPONSES ===
Response 1: <r_{1,t}>
Response 2: <r_{2,t}>
⋮
Response |S|: <r_{|S|,t}>
/no_think
    
```

3.8 Coordinate extraction

A deterministic parser ψ strips every `<think>` block from the verifier output v_t and converts each item into a pair $(\hat{\ell}^{(n)}, \hat{\mathbf{q}}^{(n)})$, where $\hat{\mathbf{q}}^{(n)} = (L, U, Z)^T$. The object coordinates are then mapped back to the original frame via $(L, U, Z) \mapsto (x, y, z) = (U, Z, -L)$, yielding

$$\hat{c}_t = \psi(v_t) = \{(\hat{\ell}^{(n)}, \hat{\mathbf{q}}^{(n)})\}_{n=1}^{\bar{M}_t}.$$

3.9 Algorithmic deterministic verifier/filter

A pair is retained when its text appears verbatim in the original scene text σ_t (to remove hallucinations that the ensemble of generators or the verifier might introduce).

$$\chi(\hat{\ell}, \hat{\mathbf{q}}; \sigma_t) = \begin{cases} 1, & \text{if } \text{regex}(\hat{\ell}, \hat{\mathbf{q}}) \subset \sigma_t, \\ 0, & \text{otherwise.} \end{cases}$$

The accepted list is therefore

$$A_t = \{(\hat{\ell}^{(n)}, \hat{\mathbf{q}}^{(n)}) \mid \chi(\hat{\ell}^{(n)}, \hat{\mathbf{q}}^{(n)}; \sigma_t) = 1\}.$$

Since σ_t is the serialized detector output, the filter enforces that every accepted item is grounded in the detected state. Consequently, the accepted set satisfies the system-level guarantee

$$A_t \subseteq D_t \quad \forall t.$$

This guarantee holds relative to the detector output and does not account for missed or mislocalized detections.

3.10 Robot action execution

The high-level object lists generated using the LLM-ensemble are executed using a single `robot_control` ROS node that

commands both UR10e arms (ROBOT1 and ROBOT2). Built on MoveIt, this node performs motion planning, trajectory streaming, and run-time safety checks in one process, thereby eliminating the latency and synchronization issues that arise when an independent planner drives each arm.

3.10.1 Control pipeline

- *Scene initialization*—At start-up, the node constructs a shared MoveIt planning scene that includes both physical arms and a kinematic “digital twin” of the opposite arm to enable local inter-robot collision checks.
- *Task ingestion*—Verified object lines (e.g., `leafcell [x y z]`) arrive on a ROS topic. They are expanded into *approach-grasp-retreat-place* primitives, whose goal poses are supplied by the YOLO/RealSense perception stream.
- *Controller type(s) and planning pipeline(s)*—velocity controller OMPL (Şucan et al., 2012); velocity limits: 20% of maximum.

For further implementation details, please refer to Shaarawy et al. (2025).

3.11 Adjustable autonomy modes

The final control interface supports variable autonomy during planning by allowing the operator to modulate the plan source at runtime. Full order (human-planned): The operator specifies a complete ordered list; the verifier and filter check it, and the motion layer executes it. Partial order (human goal, system completes): The operator names a target object or a subset of steps. The LLM infers a precedence-consistent order and inserts mandatory prerequisites. No order (system-planned): The operator states only a goal (e.g., “remove the left leafcell”); the LLM proposes a full plan subject to verification and filtering. Mid-execution override: The operator may interrupt an autonomous sequence and issue a new intent; the pipeline replans without restarting the system.

These modes map control between human and autonomy depending on operator preference and task context, which defines the variable autonomy used in this work.

4 Experiment

To utilize the framework, a series of experiments was designed to prepare, test, and verify its functionality on our disassembly setup. The goal of the first set of experiments was to select a language model that accurately and efficiently followed instructions. The goal of the second set of experiments was to verify robustness across a variety of object configurations and natural-language intents within the presented case study, using the same scene-text interface between perception and planning. The goal of the third and final set of experiments was to verify its usability in the real world by conducting a pilot study with non-expert participants.

To perform this set of experiments, we generated a dataset that worked with our previously trained weights for YOLOv8 (Jacob Solawetz, 2025). The collected dataset consisted of 200 RGB-D images, obtained using a RealSense 435i stereo

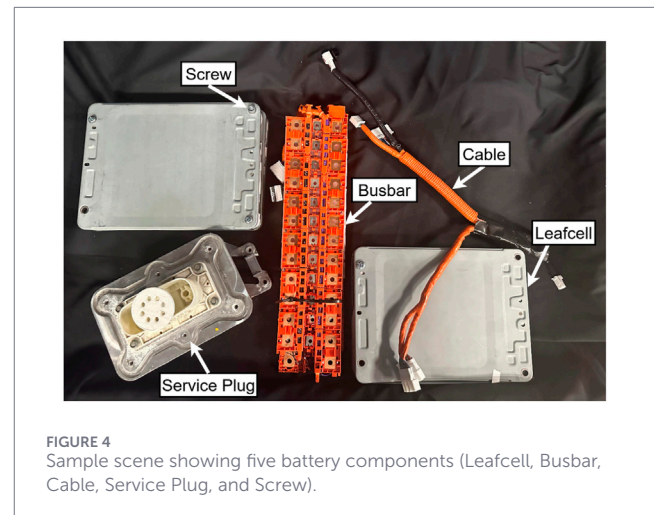


FIGURE 4 Sample scene showing five battery components (Leafcell, Busbar, Cable, Service Plug, and Screw).

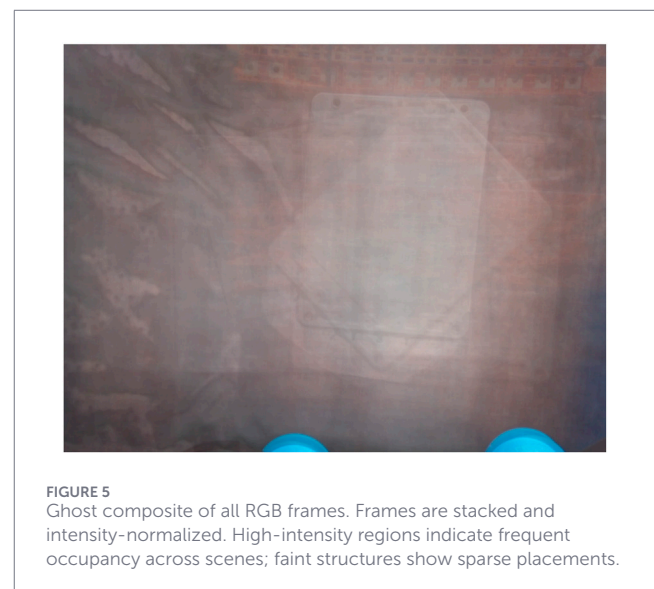


FIGURE 5 Ghost composite of all RGB frames. Frames are stacked and intensity-normalized. High-intensity regions indicate frequent occupancy across scenes; faint structures show sparse placements.

camera. All RGB images passed through our YOLOv8 weights, which provided five different classes (screw, leafcell, cable, busbar, and service plug). This 200-image dataset is fully different from the training set and was collected on a multi-robot system, under controlled conditions to keep the perception and motion predictable. The purpose of this system was to focus on evaluating the planning framework.

Figures 3, 4 show an example of object distribution and robot placement in the scene. Additionally, we analyzed the object placement distribution by stacking the RGB frames from all trials, creating a blended distribution of the objects. Figure 5 illustrates the resulting composite, revealing where objects appeared throughout the experiment. The composite provides a spatial prior over the workspace: bright areas correspond to positions of components that appear often in the dataset; dim traces reveal infrequent positions of components that would otherwise vanish in a simple average.

TABLE 1 Screening results for candidate language models (five trials per model). All models were prompted with an identical instruction set.

Model	Failures	Observation
qwen3:32b [Reasoning off]	0	Completed every trial; median latency: approximately 21 s (range 18 s–24 s). Response format is always correct, but inference speed is moderate compared with smaller models
phi4:latest	5	Failed to produce a valid plan in every trial, providing unnecessary explanations, even though runtimes remained within 30–46 s. Errors stem from systematic instruction following issues rather than timeouts
phi4-reasoning:latest	5	No valid output across all runs. Latency was highly unstable (16 s–144 s), indicating internal difficulties in reasoning steps and overthinking of instructions
phi4-reasoning:plus	3	Succeeded in the first two trials (approximately 39 s–48 s) but failed in the remaining three, yielding a 40% success rate. Failures had many varied formatting issues in the output, although speeds were relatively fast
gemma3:12b-it-qat	5	Returned malformed responses in every trial with extra verbosity (approximately 127 s–129 s). No successful completions observed
gemma3:27b-it-qat	3	Two successful runs (latencies 33 s and 70 s) but three failures caused by output format and verbosity errors, showing problems in instruction following steps; overall 40% success and high variance in turnaround time
deepseek-r1:latest	5	Exceeded the 180 s cutoff in all five trials; treated as hard failures with no usable output
qwen3:30b-a3b	3	Completed two trials (in approximately 36 s) but failed three. Failures produced verbose, off-specification answers in less than 10 s

4.1 Selection of the language model

Reliable plan generation demands an LLM that i. follows our specialized instruction-example prompt format with minimal hallucinations and ii. produces answers quickly. It must also handle underspecified user input: operators may request any order as long as the text includes at least one object in view; the LLM must then infer a feasible, precedence-consistent order and insert any mandatory prerequisites (e.g., remove screws or cables before extracting a leafcell) that the user did not mention. We screened eight open-weight LLM models offline (Table 1).

Each model received the identical user prompt U_i and system prompt θ_j . We ran five independent trials per model. A run failed if the output violated instructions (missing list, wrong delimiter, fabricated objects, or verbosity that broke the format) or exceeded 180 s. We logged the completion time for successful runs.

The model qwen3:32b completed all five trials, though with a moderate median latency of 21 s. Three models, phi4-reasoning:plus, qwen3:30b-a3b, and gemma3:27b-it-qat, succeeded in roughly 40% of runs but generated verbose or partially incorrect output in the remainder. All other checkpoints failed every trial, either timing out or ignoring mandatory constraints. Detailed outcomes are provided in Table 1.

Given these results, we selected qwen3:32b for all subsequent experiments and demonstrations, prioritizing accuracy and overall task completion rate.

4.2 Pipeline for full correctness

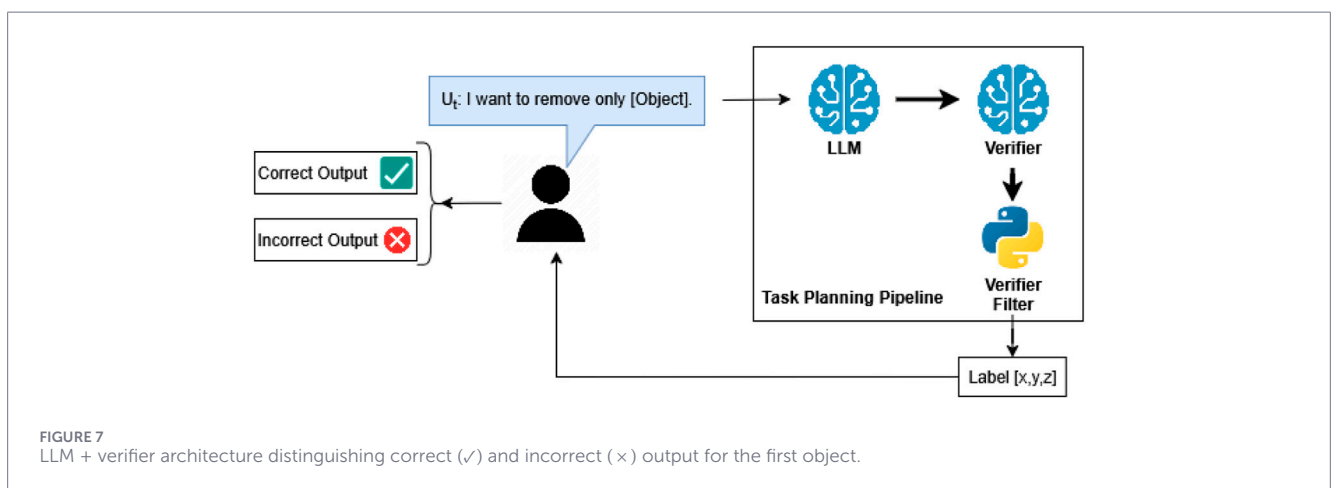
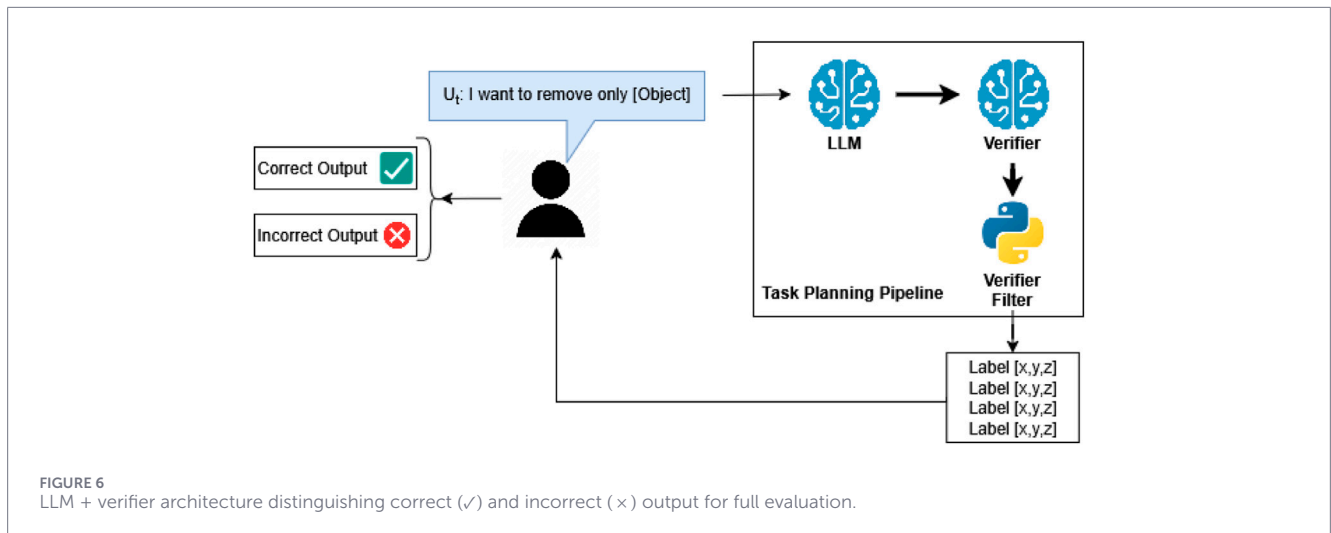
To identify the most effective pipeline configuration, a comparative evaluation was conducted across several distinct pipeline setups. Each variant employed the same underlying models and input prompts, differing only in their structural arrangement. The evaluated configurations included 1. a single-model baseline, 2. an ensemble of three models combined with a verifier, 3. an ensemble of six models combined with a verifier, 4. an ensemble of three models coupled with a verifier and an additional algorithmic verifier step, and finally, 5. an ensemble of six models integrated with a verifier and the same algorithmic verifier.

The single-model baseline consisted solely of the primary model (qwen3:32b), which was executed directly using the provided prompts without any additional validation steps. Ensemble configurations involved running multiple models concurrently, obtaining multiple candidate responses.

Subsequently, in ensemble setups, a verifier LLM took the candidate responses, systematically evaluated them according to a provided verification prompt, and returned a single, verified output. The ensemble variants marked *with Algorithmic Verifier* added a final stage where the verified responses underwent an additional data consistency check. This step retained only responses explicitly matching the original input data, effectively removing any potentially hallucinated or incorrect information.

All evaluations were conducted using 600 user-generated prompts across 200 distinct real robot scenarios (3 per scenario). The responses were evaluated by operators, who determined whether the output met the logical requirements of the task while adhering to its given intent. Figure 6 shows an example of an expected output from a “full” trial, where an ordered list of objects is likely to be the outcome.

The data from the scenarios and each user prompt were recorded, and the tasks were repeated three times for each of the evaluated configurations. Each output was then assessed under consistent conditions and prompts, ensuring fairness and comparability across pipeline variants. The primary metrics recorded included the correctness of outputs, execution time per pipeline configuration, and adherence to instruction sets. In total, 9,000 experiments were conducted across the five configurations, with the hypothesis that the model with the largest ensemble and both verifiers would yield higher accuracy than the single-model configuration.



4.3 Pipeline for next object correctness

To further evaluate the effectiveness of each pipeline configuration, we performed a secondary analysis on the previously obtained experimental data. In this evaluation, we assessed only whether the first object in each generated response correctly matched the next object to be removed, according to the task logic and intent. The goal of this targeted evaluation was to determine if ensemble and verifier mechanisms specifically improve the initial selection within generated action plans. The verification flow for first-object accuracy is summarized in Figure 7.

The same five pipeline configurations as in the “full” correctness trials were compared. No new experiments were conducted; instead, previously recorded outputs from the original 9,000 trials (600 user-generated prompts across 200 scenarios, repeated three times per configuration) were re-evaluated against this narrower correctness criterion. Figure 6 shows an example of an expected output from a “next object correctness” trial, where, from the ordered list, only the first object is kept in the output.

The primary metric recorded was the accuracy of the first suggested object in the response, specifically assessing whether it

matched the correct object designated for immediate removal. This evaluation enabled a direct comparison of pipeline configurations to determine whether the ensemble and verifier steps provided a clear advantage over the single-model baseline, particularly in terms of the accuracy of initial task predictions.

4.4 Real-world usability experiment

We conducted a real-world usability pilot experiment to assess how non-experts issue intent-driven disassembly commands using our LLM-ensemble pipeline. Seven volunteers ($n = 7$), with no prior experience controlling robots, interacted with the system via a dedicated front end. The goal was to evaluate operator user experience and workload when commanding multi-robot disassembly via natural language.

Each participant issued two intents (one per arm) in plain language. The pipeline then generated an ordered list per arm that respected the disassembly precedence (without the need to provide all objects of the intended disassembly explicitly).

After a brief, single training session covering scene representation, UI elements, and camera views, participants were instructed to operate a two-arm disassembly system by specifying

what to remove or disassemble (one natural-language intent per arm). They observed two example intents during training (e.g., “remove only the leafcell from the left module”) and then wrote their own intents for each scene. We evaluated a single condition: the full pipeline (selected backbone LLM with the 6-LLM ensemble, verifier, and final consistency filter) integrated with two UR10e arms; no alternative planners or interfaces were tested here (comparisons of the pipeline appear in the offline object correctness experiments), but a 300 s baseline of an expert operator performing the task without LLM assistance was recorded.

Participants reviewed each object appearing in the camera views and in the list of possible objects (screw, leafcell, cable, busbar, and service plug) before taking any action.

The procedure for each trial experiment was as follows: 1. press *Capture* to run YOLOv8 and generate the structured text scene and 3D view; 2. type one intent per arm; 3. press *Let LLM plan* to obtain the removal lists; 4. verify that the lists matched the desired human intent and press *Go* to execute; and 5. after execution, report whether each action reflected the true intent and complete the NASA-TLX questionnaire. The primary measures were time-to-action and perceived workload (NASA-TLX); secondary measures included plan acceptance rate and execution success.

For each participant, we recorded the following: i. *LLM plan success* (Yes/No) for each arm based on participant confirmation of their intention; ii. *time-to-action* (seconds until pressing “Go” with a correct plan); and iii. *NASA-TLX workload questionnaire*.

To contextualize time-to-action, we used a manual baseline from an expert operator. The operator ran YOLOv8, reviewed the detections, and hand-entered coordinates according to his own disassembly plan for three items from the YOLO position list. He then executed the plan (“Go”). The mean over three runs was 300 s. This baseline is not a paired control; it is only for interpreting the pipeline times.

After completing the NASA-TLX questionnaire, the participants were allowed to issue extra commands for exploration; these interactions were not logged as part of the study dataset. Participants informally reported that the system was very simple to use and that they could have completed the trials without the training session. Some noted that planning also worked with non-English requests during their exploration attempts.

5 Results

This section reports three things. First, a brief analysis of the language used in the operator requests to form the dataset is presented. Second, a quantitative study of the planning pipeline under various configurations is conducted, comparing different ablations of the system. Third, the workload analysis and real-world assessment results are provided.

For the pipeline quantitative study, we evaluate two targets: i. *full correctness* (the full removal set in correct order) and ii. *next object correctness* (the first object only). For both targets, we report accuracy and time (average and median) per stage and ensemble size.

5.1 Language results

We start with the request language because it sets the operating context for the planner. The instructions given by the operator shape intent and action classes, while the presence of specific object names sets the scope of the feasible plans. Figure 8 summarizes the most frequent terms and object references.

The dominant words appearing on the requests given by the operators were *want* and *remove*, which align with intent specification (*want*) and the disassembly task (*remove*). Object terms (*leafcell*, *service plug*, *busbar*, *screw*, and *cable*) appear high in the list, with *leafcell* being the most mentioned object and *cable* the least.

Numerically, aggregating singular and plural mentions, the *cable* is mentioned more than 90 times, *screws* more than 100, *service plugs* more than 120, *busbars* more than 125, and *leafcells* more than 135. Generic selectors such as *all*, *objects*, and *everything* were used more than 70 times across the 600 requests. These frequencies describe the request mix that the ablations evaluate. These do not change the scoring rules but explain the prevalence of certain object combinations and the length of outputs in the test cases.

5.2 Pipeline results

We evaluated the pipeline on an operator-prepared fixed dataset of scenes and prompts, different from the free-form “play” with non-expert participants. For each scene, we defined one or more admissible ground-truth removal sequences that satisfy disassembly precedence. The LLM received an intent prompt that could be a full or partial order or only a target object; the only requirement was that the operator prompt reference at least one object in the current scene. The LLM was allowed to reorder objects and to insert mandatory prerequisites (e.g., removing screws or cables before a leafcell) that the user did not explicitly request.

Accuracy was computed using two objective metrics. Full-sequence correctness: the predicted ordered list exactly matches any admissible ground-truth sequence. Next-object correctness: the first predicted object is the correct next action, regardless of the remaining list. Timing uses the same executions for both analyses. Therefore, the average and median times are reported once per configuration and shared across the full and next evaluations.

In Table 2, the accuracies for the different configurations of 1 LLM, 3 LLMs ensemble + verifier (Ensemble), and ensemble + verifier + algorithmic verifier (Final), along with 6 LLMs ensemble + verifier (Ensemble) and ensemble + verifier + algorithmic verifier (Final), are reported, for both *full correctness* and *next object correctness* trials, accompanied by the respective time of their calculations. For a clearer observation and easier comparison of the accuracy gains, Figure 9 presents the trendline for both sets of trials across the various pipeline configurations.

Figure 10 displays a violin plot showing the distribution of time per request across the dataset, and Figure 11 complements this by reporting the distribution of final-stage times for the 9,000 runs, with density values for each of the time bandwidths.

Figure 12 reports per-stage processing times for the full evaluation. The left panel displays the mean time per request, while the right panel shows the median time per request. The x-axis lists stages (LLM, Ensemble, and Final). Within each stage, bars

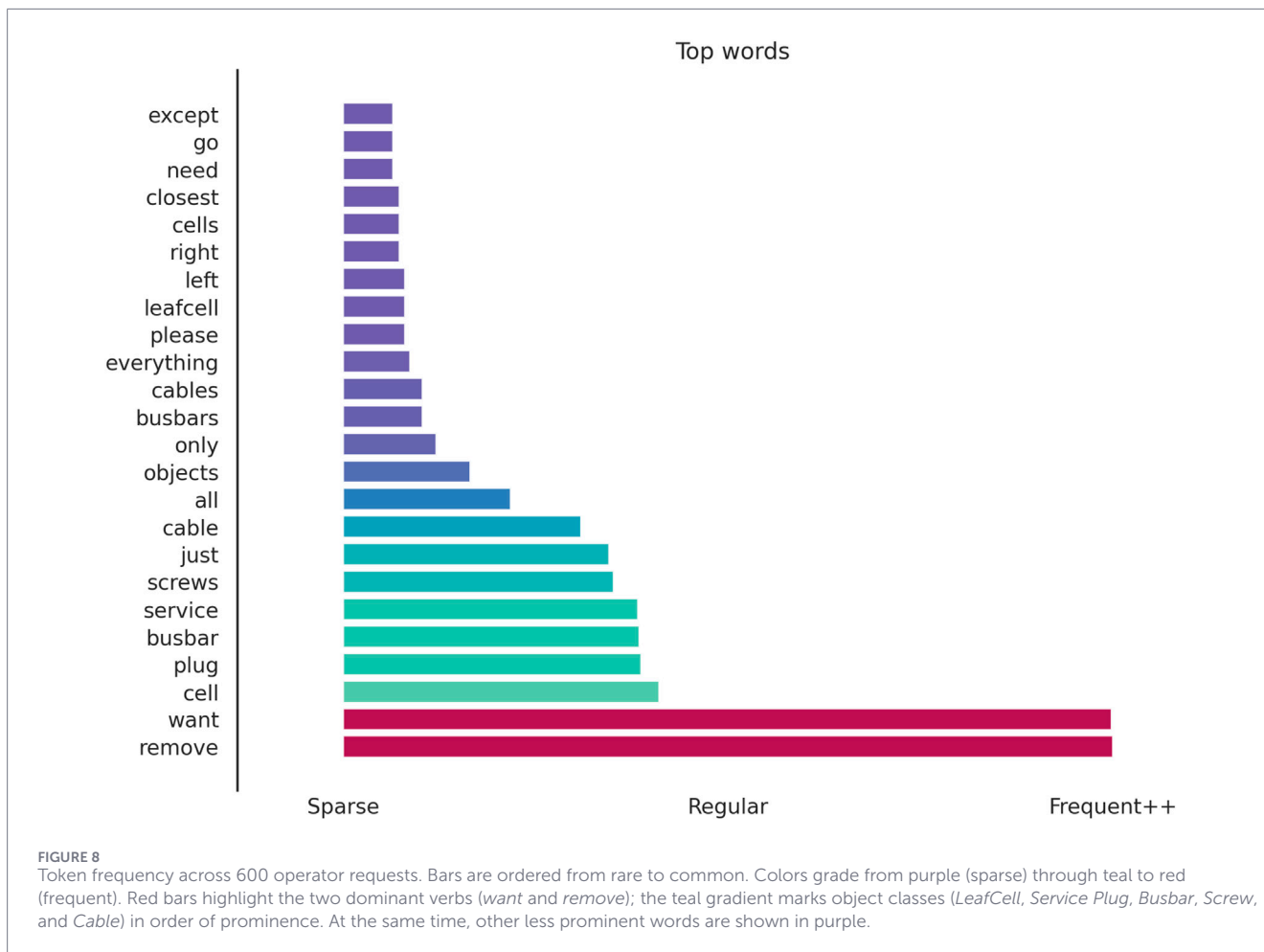


TABLE 2 Pipeline results from the same executions. Accuracy for Full and Next analyses, with shared average and median times per configuration.

LLMs	Stage	Accuracy		Time (s)	
		Full	Next	Avg	Med
1	LLM	0.761	0.866	5.571	4.260
3	Ensemble	0.787	0.863	21.934	19.250
3	Final	0.796	0.871	22.153	19.300
6	Ensemble	0.816	0.888	32.504	28.935
6	Final	0.824	0.894	32.744	29.005

correspond to the 1-LLM baseline for LLM and to the 3-LLM and 6-LLM configurations for Ensemble and Final. The y-axis is measured in seconds, measured at the time the last word is output.

Figure 13 plots payload size (tokens used) against final-stage latency for the full evaluation. The left panel uses input token counts; the right panel uses output token counts. Each point represents a single request from the same executions as in Table 2. The model tokenizer measures tokens; latency is the time recorded for the final stage. The separate series indicate the 3-LLM and 6-LLM ensembles.

Table 3 lists *p*-values for pairwise stage comparisons (LLM vs. Ensemble, LLM vs. Final, and Ensemble vs. Final) under the full (all objects) and next (next object) settings for 3-LLM and

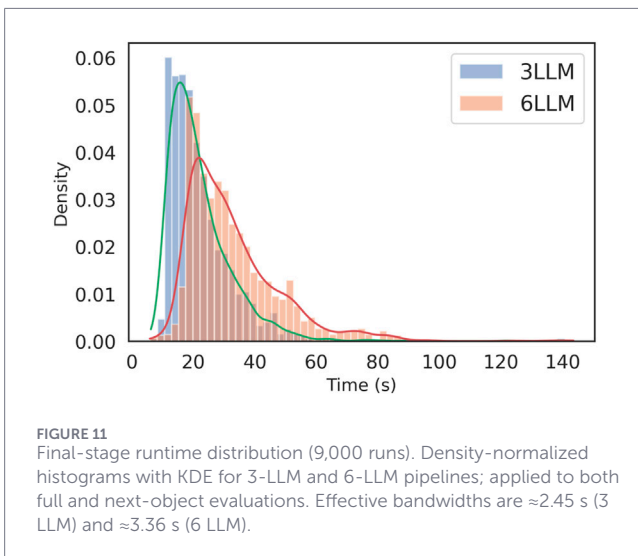
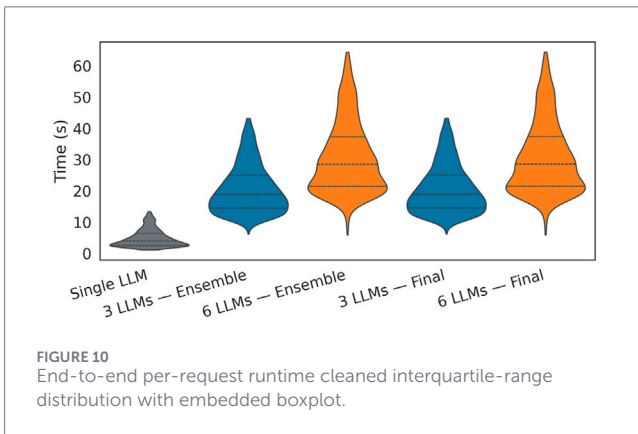
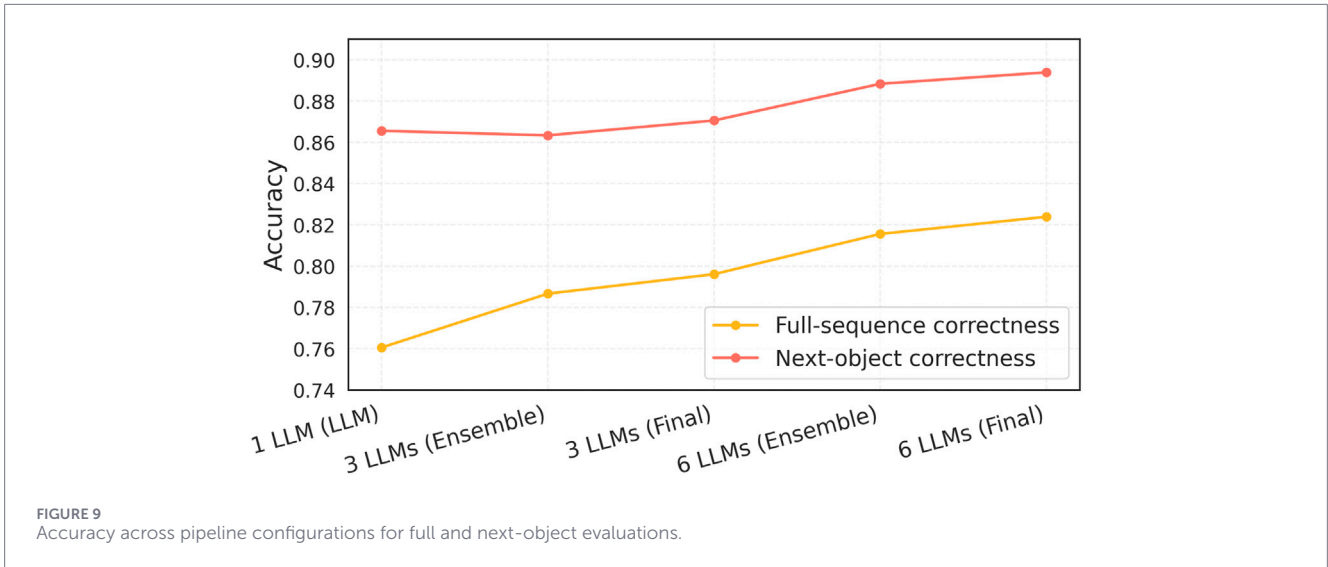
6-LLM configurations. Entries with a *p*-value less than 0.05 are marked with a checkmark. We used two-sided paired t-tests on per-prompt accuracy to compare stages, leveraging the fact that the same prompts were evaluated at each stage to control for between-prompt variability. This test assumes independence across prompts and approximate normality of the paired differences, with large *N* (1,800). The table reports *p*-values only.

Note that prompts could be partial; the LLM often inserted necessary prerequisites (e.g., screws or cables) before the requested target to satisfy precedence while still matching the operator’s intent under our scoring rules.

5.3 Real-world pilot results

Seven volunteers (*n* = 7) completed one session each. The study protocol, including all operator-in-the-loop experiments, was approved by the University of Birmingham Ethics Committee under project ERN_3337. Each participant issued two intents (one per arm), totaling 14 intended actions. All disassembly actions were successfully planned as intended by the participants through the framework. Success was achieved for every participant in both cases in 14/14 actions.

Table 4 reports the mean and median completion times for the participants using the pipeline to control the robots. A 300-s manual baseline serves as a contextual reference from an expert operator,



performing the task without LLM-assisted planning, against which these data are compared.

Table 5 summarizes the global NASA-TLX workload (0–100), which was used to report the overall perceived workload for the session.

Table 6 reports the mean \pm SD and median for each NASA-TLX subscale (0–100): mental demand, physical demand, temporal demand, performance, effort, and frustration.

6 Discussion

This work tested intent-driven ensemble LLM planners, performing a case study on EV-battery component disassembly using two UR10e arms. However, the architecture is modular and, in principle, can be transferred to other cases by swapping the perception model and motion backend, provided the input structure is maintained.

6.1 Model selection

As the model selection and preliminary experiments show, the most important characteristic when choosing among low-parameter-count models was their ability to adhere to instructions. The most common errors found in models that did not adhere to the instructions were making formatting errors or being overly verbose in explaining their conclusions before providing the list of components, even though the instructions explicitly requested only the list as output. Qwen3-32B (reasoning off) provided the best responses, although its speed remained above 20 s on average.

6.2 Pipeline correctness

Across the 9,000 runs (Table 2), full object correctness increased from 0.761 (single model) to 0.816 (6-model + verifier) and to 0.824 with the final algorithmic filter. That is a +6.3 percentage-point absolute gain over the single model or $\approx +8.3\%$ relative gain. Accuracy increases with ensemble size (Figure 9). The deterministic filter adds small but consistent increments (0.9 pp for 3-LLM and 0.8 pp for 6-LLM) while removing hallucinated objects at nearly

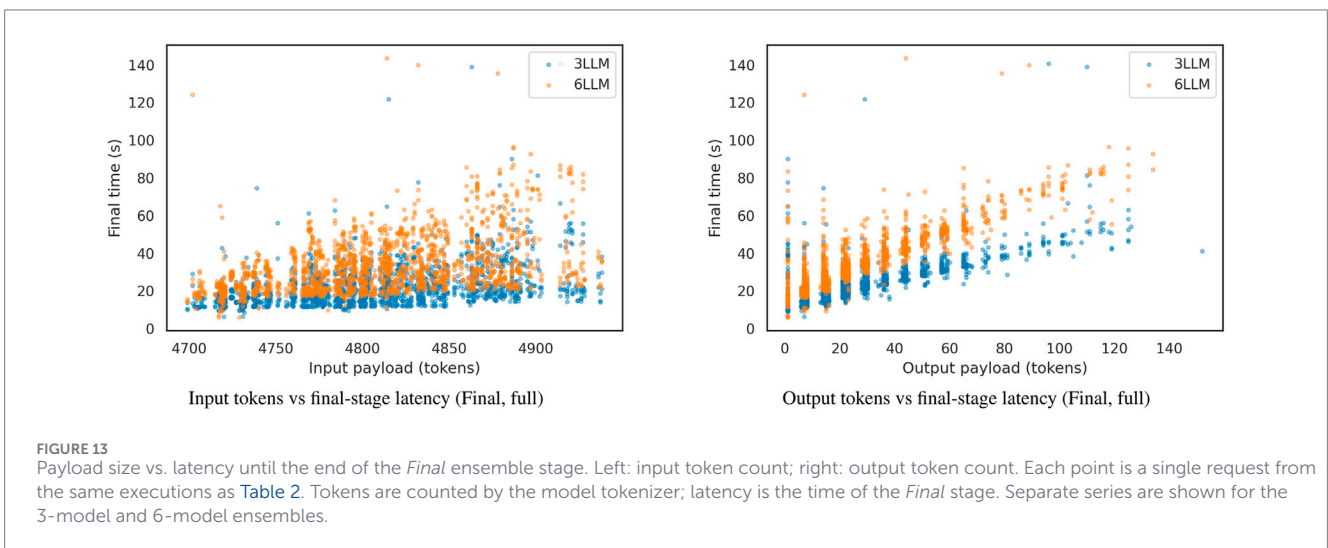
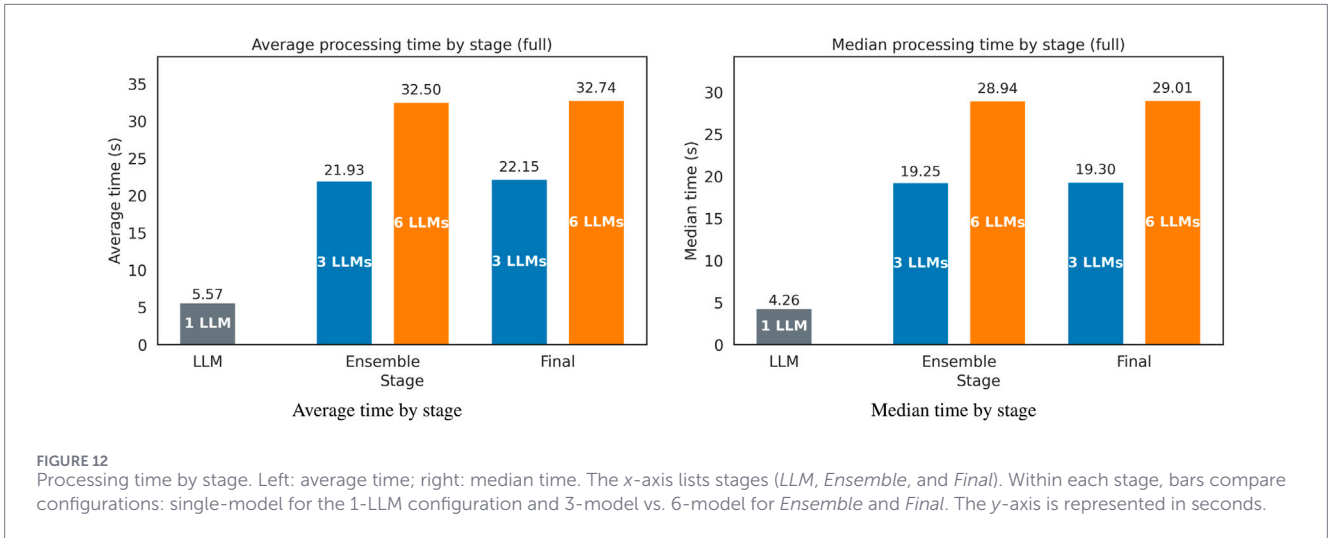


TABLE 3 Stage comparisons using paired t-tests on matched 0/1 correctness (two-sided). A checkmark indicates $p < 0.05$.

Comparison	Setting	3LLM p	Sig.@0.05	6LLM p	Sig.@0.05
LLM vs. ensemble	Full	2.19×10^{-7}	✓	1.57×10^{-9}	✓
	Next	2.90×10^{-2}	✓	1.13×10^{-3}	✓
LLM vs. final	Full	6.61×10^{-10}	✓	1.19×10^{-11}	✓
	Next	1.45×10^{-3}	✓	7.70×10^{-5}	✓
Ensemble vs. final	Full	3.61×10^{-5}	✓	1.74×10^{-3}	✓
	Next	7.78×10^{-4}	✓	2.68×10^{-3}	✓

zero time cost. Paired t-tests on matched per-prompt accuracy (two-sided) confirm significant improvements from single-model to ensemble and to the final all-integrated framework (Full: 3-LLM $p = 2.19 \times 10^{-7}$, 6-LLM $p = 1.57 \times 10^{-9}$; LLM \rightarrow Final: 3-LLM $p = 6.61 \times 10^{-10}$, 6-LLM $p = 1.19 \times 10^{-11}$). Ensemble \rightarrow Final yields small but significant gains (Full: 3-LLM $p = 3.61 \times 10^{-5}$, 6-LLM $p = 1.74 \times 10^{-3}$; see Table 3).

For next-object correctness, the single model yields a high accuracy (0.866). The 3-LLM configuration achieves 0.863 (Ensemble) and 0.871 (Final), while the 6-LLM configuration reaches 0.888 (Ensemble) and 0.894 (Final), indicating an increasing trend with a slight dip at 3-LLM before increasing at 6-LLM. Paired t-tests show significant gains over the single model (3-LLM: LLM \rightarrow Ensemble $p = 2.90 \times 10^{-2}$, LLM \rightarrow Final $p = 1.45 \times 10^{-3}$; 6-LLM: $p =$

TABLE 4 Completion time (pipeline condition). Values are presented as the mean \pm SD and median in seconds.

Metric	Mean \pm SD (s)	Median (s)
Time to action	197.86 \pm 19.61	195.00

TABLE 5 Global NASA-TLX workload (0–100). Values are presented as the mean \pm SD and median.

Metric	Mean \pm SD	Median
NASA-TLX (global)	14.95 \pm 11.80	9.67

TABLE 6 NASA-TLX subscales (0–100): mean \pm SD and median across participants.

Subscale	Mean \pm SD	Median
Mental demand	10.00 \pm 5.77	10.00
Physical demand	7.86 \pm 5.67	5.00
Temporal demand	7.86 \pm 5.67	5.00
Performance	27.86 \pm 34.03	10.00
Effort	10.00 \pm 5.00	10.00
Frustration	10.00 \pm 6.46	5.00

1.13×10^{-3} and 7.70×10^{-5}). Ensemble \rightarrow Final adds further small but significant improvements (3-LLM $p = 7.78 \times 10^{-4}$; 6-LLM $p = 2.68 \times 10^{-3}$; see Table 3).

Latency increases with ensemble size: mean end-to-end time is 5.57 s (single model), 21.93–22.15 s (3-model, \pm filter), and 32.50–32.74 s (6-model, \pm filter) (Table 2). Median times show the same pattern (4.26 s \rightarrow 19.25–19.30 s \rightarrow 28.94–29.01 s). The final filter adds approximately 0.2 s on average, which is negligible compared to ensemble sampling and verification. Distributions are right-tailed (Figure 10) and widen with ensemble size, with the final-stage histograms (Figure 11) showing the bandwidth increasing from ≈ 2.45 s (3-model) to ≈ 3.36 s (6-model). The whole pipeline's mean latency scales to approximately 5.9 times the single-model baseline (+488% relative), yielding a respectable and expected compute-accuracy trade-off. For a planning pipeline, although these numbers are slower than those of the pre-planned methods or distance-based planning, they are suitable for tasks that require some adaptability and more intelligent, dynamic planning capabilities under different conditions. Token-latency in Figure 13 suggests a weak positive correlation between payload and final-stage time, with most of the latency changes in the spread dominated by ensemble size rather than token counts.

The measured accuracy lift provided by the deterministic filter is small but consistent across both metrics. It comes with a mean time overhead of approximately 0.20 s (Figure 12), acting as a very effective low-cost last line of defense against hallucinations. For our framework, the scene included five component classes (leafcell, busbar, cable, service plug, and screw; Figure 4) detected using YOLOv8. Although the LLM was never given explicit labels, it was able to adapt properly to the components of the disassembly

task using the provided instructions, supporting the claim that the planning layer is largely label-set agnostic when the scene text schema is preserved (with no model working in preliminary tests with unstructured data).

The bar plot of prompt tokens (Figure 8) shows that operator intents are dominated by “want” and “remove,” due to the disassembly nature of the trials, with object mentions led by “leafcell” and “busbar.” Generic selectors, such as “all/objects/everything,” appear frequently, explaining plans that contained a high number of objects and why precedence checks matter in disassembly tasks.

6.3 Human pilot experiment

The results of the pilot suggest that a language-level variable autonomy mechanism with verification maintains low operator workload while preserving user control.

First, all 14 intents were executed only after participants reviewed and approved the candidate lists. The final execution remained operator-controlled. Second, the perceived workload was low (Nikulin et al., 2019; Grier, 2015) (NASA-TLX mean 14.95 and median 9.67) after a brief single training session (Tables 4–6). The UI only displayed the detected objects by YOLO in the form of a 2D image and a 3D object display. Participants did not tune any low-level motion or grasp parameters; they only issued intents and approved an ordered list. This likely reduced the cognitive load. Third, time-to-action averaged 197.86 s (median 195 s), lower than a contextual manual reference of ≈ 300 s, indicating that plan synthesis plus checking added little overhead relative to manual planning.

The verifier LLM and the final algorithmic filter removed format and consistency errors before execution, limiting cognitive effort to intent expression and a single “Go” confirmation. The sample is small and lacks an interface control, but the pattern aligns with variable autonomy: the autonomy proposes and checks plans, and the operator authorizes execution.

7 Conclusion

This work presented an intent-driven planning pipeline for multi-robot collaboration and disassembly that integrates perception-to-text scene encoding, an LLM ensemble sampled from a single checkpoint, a format/priority LLM verifier, and a final deterministic consistency filter wired end-to-end.

An ensemble-with-verification architecture improves multi-robot disassembly planning over a single LLM. On 200 real scenes and 600 intents, full-sequence accuracy increases from 0.761 to 0.824 with a 6-LLM ensemble plus an LLM verifier and a final deterministic filter; next-object accuracy increases from 0.866 to 0.894. Under paired *t*-tests, gains are significant, and the deterministic filter provides additional small, significant, safety-oriented improvements at negligible time cost. Latency scales with ensemble size ($\approx 5.9 \times$ the single-model baseline), reflecting the standard accuracy–compute trade-off.

In real-world use, non-experts could issue intents that were executed as intended after list review, with a low perceived workload (≈ 15) and shorter time-to-action than with a contextual manual

reference (300 s). Time-to-action was faster than the manual baseline by roughly one-third on average ($\approx 34\%$ faster), with the median approximately 35% faster and run-to-run variability $\pm 10\%$ relative to the mean.

The LLM ensemble, equipped with explicit verification and data consistency checks, results in a viable approach to injecting reliability and adaptability into intent-driven planning for collaborative disassembly.

7.1 Limitations and future work

Our ensemble improves plan correctness but shows the expected accuracy–latency trade-off: accuracy gains taper, while the mean latency increases from 5.57 s (single model) to 32.74 s (6-LLM) on the same prompts. Future work will incorporate dynamic computation, including early-exit voting when candidates agree and input-complexity gating between 1/3/6 models. We also aim to cache and reuse verified partial plans for recurring intents (when possible). We also do not benchmark end-to-end VLM/VLA action policies on our disassembly cell; doing so would require domain adaptation and safety validation beyond the scope of this study.

Verification is text-only today (format/precedence rules plus a data consistency filter). This prevents many errors but cannot enforce geometric feasibility. We will add geometry-aware checks (reachability, collisions, and graspability) to reject impossible steps and to provide minimal counterexamples that guide operator edits. The pipeline currently assumes reliable detections but is highly dependent on the point cloud and the YOLOv8 model weights. To improve the detection rates, we will incorporate multi-view fusion and uncertainty-aware perception to enhance its accuracy. A pre-grasp re-verification stage with live sensing and per-action risk scores (e.g., hazard proximity or collision) can enable safe local reordering within the stated intent.

Diversity in our results stems from one checkpoint with different seeds. Mixing model families and prompt variants may outperform the same-checkpoint ensembles at a similar computational cost, which is also worth investigating as part of future work. Although the verifier prompt is domain-agnostic, we evaluated one domain with five component classes. We will evaluate other assembly/disassembly settings (e-waste, other types of end-of-life product dismantling, and nuclear decommissioning). Another area for deeper study in future work is expanding the human user study to include larger numbers of participants and exploring whether and how human-perceived workload varies across different types of tasks and environments.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Ethics statement

The studies involving humans were approved by the Research Ethics Team of the University of Birmingham (Sue Cottam:

Research Ethics Manager, Sam Waldron: Deputy Research Ethics Officer). The studies were conducted in accordance with the local legislation and institutional requirements. The participants provided their written informed consent to participate in this study.

Author contributions

CE: Conceptualization, Data curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review and editing. CAC: Conceptualization, Data curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review and editing. AR: Conceptualization, Funding acquisition, Investigation, Project administration, Resources, Supervision, Writing – original draft, Writing – review and editing. MC: Formal Analysis, Funding acquisition, Project administration, Software, Supervision, Writing – original draft, Writing – review and editing. RS: Funding acquisition, Project administration, Resources, Supervision, Writing – original draft, Writing – review and editing.

Funding

The author(s) declared that financial support was received for this work and/or its publication. This work was funded by the project called “Research and Development of a Highly Automated and Safe Streamlined Process for Increasing Lithium-ion Battery Repurposing and Recycling” (REBELION) under Grant 10079049 and partially supported by the Ministry of National Education, Republic of Turkey.

Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declared that generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher’s note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of

their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

- Agency (2025). Supporting the implementation of robotics and remote systems in the nuclear back-end. Tech. Rep. NEA/RWM/R(2024)3. Paris, France: OECD Nuclear Energy Agency NEA. Available online at: https://www.oecd-nea.org/jcms/pl_109733/upload/docs/application/pdf/2025-09/rwm_r_2024_3_2025-09-16_17-21-55_700.pdf (Accessed December 19, 2025).
- Asif, M. E., Rastegarpanah, A., and Stolkin, R. (2024). Robotic disassembly for end-of-life products focusing on task and motion planning: a comprehensive survey. *J. Manuf. Syst.* 77, 483–524. doi:10.1016/j.jmsy.2024.09.010
- Belsare, A., Karimi, Z., Mattson, C., and Brown, D. S. (2025). “Toward zero-shot user intent recognition in shared autonomy,” in *Proceedings of the 2025 ACM/IEEE international conference on human-robot interaction (HRI)* (ACM/IEEE).
- Bernardo, R., Sousa, J. M., and Gonçalves, P. J. (2023). A novel framework to improve motion planning of robotic systems through semantic knowledge-based reasoning. *Comput. and Industrial Eng.* 182, 109345. doi:10.1016/j.cie.2023.109345
- Bird, B., Nancekievill, M., West, A., Hayman, J., Ballard, C., Jones, W., et al. (2021). Vega—a small, low cost, ground robot for nuclear decommissioning. *J. Field Robotics* 39, 232–245. doi:10.1002/rob.22048
- Bowman, M., Zhang, J., and Zhang, X. (2024). Intent-based task-oriented shared control for intuitive telemanipulation. *J. Intelligent and Robotic Syst.* 110, 167. doi:10.1007/s10846-024-02185-1
- Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choremanski, K., et al. (2023). RT-2: vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818. doi:10.48550/arXiv.2307.15818
- Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004). “Ensemble selection from libraries of models,” in *Proceedings of the twenty-first international conference on machine learning*, 18.
- Chiou, M., Hawes, N., and Stolkin, R. (2021). Mixed-initiative variable autonomy for remotely operated mobile robots. *ACM Trans. Human-Robot Interact. (THRI)* 10, 1–34. doi:10.1145/3472206
- Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., et al. (2022). Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416. doi:10.48550/arXiv.2210.11416
- Contreras, C. A., Chiou, M., Rastegarpanah, A., Szulik, M., and Stolkin, R. (2025a). Probabilistic human intent prediction for mobile manipulation: an evaluation with human-inspired constraints. arXiv preprint arXiv:2507.10131.
- Contreras, C. A., Rastegarpanah, A., Chiou, M., and Stolkin, R. (2025b). A mini-review on mobile manipulators with variable autonomy. *Front. Robotics AI* 12, 1540476. doi:10.3389/frobt.2025.1540476
- Crandall, J. W., and Goodrich, M. A. (2001). “Experiments in adjustable autonomy,” in *2001 IEEE international conference on systems, man and cybernetics. e-Systems and e-Man for cybernetics in cyberspace (cat. No. 01CH37236)* (IEEE), 3, 1624–1629. doi:10.1109/icsmc.2001.973517
- Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., et al. (2023). PaLM-E: an embodied multimodal language model. arXiv preprint arXiv:2303.03378. doi:10.48550/arXiv.2303.03378
- Dussolle, A., Díaz, A. C., Sato, S., and Devine, P. (2025). “M-ifeval: multilingual instruction-following evaluation,” in *Findings of the association for computational linguistics: NAACL 2025* (Albuquerque, USA: Association for Computational Linguistics).
- Erdogan, C., Contreras, C. A., Stolkin, R., and Rastegarpanah, A. (2024). Multi-robot task planning for efficient battery disassembly in electric vehicles. *Robotics* 13, 75. doi:10.3390/robotics13050075
- Fatemi-Anaraki, S., Tavakkoli-Moghaddam, R., Foumani, M., and Vahedi-Nouri, B. (2023). Scheduling of multi-robot job shop systems in dynamic environments: mixed-integer linear programming and constraint programming approaches. *Omega* 115, 102770. doi:10.1016/j.omega.2022.102770
- Fedus, W., Zoph, B., and Shazeer, N. (2021). Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:2101.03961. doi:10.48550/arXiv.2101.03961
- Garcia, R., Chen, S., and Schmid, C. (2025). Towards generalizable vision-language robotic manipulation: a benchmark and llm-guided 3d policy
- Grier, R. A. (2015). How high is high? a meta-analysis of nasa-tlx global workload scores. *Proc. Human Factors Ergonomics Society Annual Meeting* 59, 1727–1731. doi:10.1177/1541931215591373
- Jacob Solawetz, F. (2025). What is yolov8? the ultimate guide. Available online at: <https://blog.roboflow.com/whats-new-in-yolov8/> (Accessed July 2, 2025).
- Jiang, Y., Wang, Y., Zeng, X., Zhong, W., Li, L., Mi, F., et al. (2024). “Followbench: a multi-level fine-grained constraints following benchmark for large language models,” in *Proceedings of the 62nd annual meeting of the association for computational linguistics (ACL)* (Bangkok, Thailand: Association for Computational Linguistics).
- Kim, C., Lee, C., and Mutlu, B. (2024). Understanding large-language model (LLM)-powered human-robot interaction. *arXiv*, 371–380. doi:10.1145/3610977.3634966
- Kolling, A., Walker, P., Chakraborty, N., Sycara, K., and Lewis, M. (2016). Human interaction with robot swarms: a survey. *IEEE Trans. Human-Machine Syst.* 46, 9–26. doi:10.1109/THMS.2015.2480801
- Lakhnati, Y., Pascher, M., and Gerken, J. (2024). Exploring a gpt-based large language model for variable autonomy in a vr-based human-robot teaming simulation. *Front. Robotics AI* 11, 1347538. doi:10.3389/frobt.2024.1347538
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Advances in neural information processing systems (NeurIPS)* (Red Hook, NY: Curran Associates, Inc.).
- Li, X., Zhang, M., Geng, Y., Geng, H., Long, Y., Shen, Y., et al. (2024). “Maniplm: embodied multimodal large language model for object-centric robotic manipulation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (IEEE/CVF).
- Liu, C., Hamrick, J. B., Fisac, J. F., Dragan, A. D., Hedrick, J. K., Sastry, S. S., et al. (2016). “Goal inference improves objective and perceived performance in human-robot collaboration,” in *Proceedings of the 2016 international conference on autonomous agents and multiagent systems (AAMAS)* (Singapore: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS)).
- Liu, H., Zhu, Y., Kato, K., Tsukahara, A., Kondo, I., Aoyama, T., et al. (2024). Enhancing the llm-based robot manipulation through human-robot collaboration. *IEEE Robotics Automation Lett.* 9, 6904–6911. doi:10.1109/Lra.2024.3415931
- Long, Z., Killick, G., McCreadie, R., and Camarasa, G. A. (2024). Robollm: robotic vision tasks grounded on multimodal large language models
- Marturi, N., Rastegarpanah, A., Takahashi, C., Adjigle, M., Stolkin, R., Zurek, S., et al. (2016). “Towards advanced robotic manipulation for nuclear decommissioning: a pilot study on tele-operation and autonomy,” in *2016 international conference on robotics and automation for humanitarian applications (RAHA)* (IEEE), 1–8. doi:10.1109/RAHA.2016.7931866
- Marturi, N., Rastegarpanah, A., Rajasekaran, V., Ortenzi, V., Bekiroglu, Y., Kuo, J. A., et al. (2017). “Towards advanced robotic manipulations for nuclear decommissioning,” in *Robots operating in hazardous environments*. Editor H. Canbolat (London, UK: InTech), 61–83. doi:10.5772/intechopen.69739
- Methnani, L., Chiou, M., Dignum, V., and Theodorou, A. (2024). Who’s in charge here? a survey on trustworthy ai in variable autonomy robotic systems. *ACM Computing Surveys* 56, 1–32. doi:10.1145/3645090
- Moshayedi, A. J., Mohamad, S., Hamid, N., Ahmad, M. S., Riaz, S. M., Majlessi, S. E. A. A., et al. (2024a). Micro robots in healthcare: features, applications, and future directions: a feature review. *EAI Endorsed Trans. AI Robotics* 3, 10.4108/airo.5602
- Moshayedi, A. J., Roy, A. S., Eftekhari, S. A., Liao, L., and Kolahdooz, A. (2024b). Design and development of FOODIEBOT robot: from simulation to design. *IEEE Access* 12, 36148–36172. doi:10.1109/ACCESS.2024.3355278
- Moshayedi, A. J., Xu, D., Sharifdoust, M., Khan, A. S., Khan, Z. H., and Andani, M. E. (2025). Comparative performance analysis of a novel fusion-based algorithm for AGV navigation. *Adv. Comput. Intell.* 5, 5. doi:10.1007/s43674-025-00083-z
- Nikulin, C., Lopez, G., Piñonez, E., Gonzalez, L., and Zapata, P. (2019). Nasa-tlx for predictability and measurability of instructional design models: case study in design methods. *Educ. Technol. Res. Dev.* 67, 467–493. doi:10.1007/s11423-019-09657-4
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., et al. (2022). “Training language models to follow instructions with human feedback,” in *Advances in neural information processing systems (NeurIPS)* (Red Hook, NY: Curran Associates, Inc.).
- Pan, T., Verginis, C. K., and Kavrakli, L. E. (2024). “Robust and safe task-driven planning and navigation for heterogeneous multi-robot teams with uncertain dynamics,” in *2024 IEEE/RISJ international conference on intelligent robots and systems (IROS)*, 3482–3489. doi:10.1109/IROS58592.2024.10802695

- Panagopoulos, D., Petousakis, G., Stolkin, R., Nikolaou, G., and Chiou, M. (2021). "A bayesian-based approach to human operator intent recognition in remote Mobile robot navigation," in *2021 IEEE international conference on systems, man, and cybernetics (SMC)* (IEEE), 125–131.
- Raposo, D., Ritter, S., Richards, B., Lillicrap, T., Humphreys, P. C., and Santoro, A. (2024). Mixture-of-depths: dynamically allocating compute in transformer-based language models.
- Reinmund, T., Salvini, P., Kunze, L., Jirotko, M., and Winfield, A. F. (2024). Variable autonomy through responsible robotics: design guidelines and research agenda. *ACM Trans. Human-Robot Interact.* 13, 1–36. doi:10.1145/3636432
- Scerri, P., Pynadath, D. V., and Tambe, M. (2002). Towards adjustable autonomy for the real world. *J. Artif. Intell. Res.* 17, 171–228. doi:10.1613/jair.1037
- Shaarawy, A., Erdogan, C., Stolkin, R., and Rastegarpanah, A. (2025). Multi-robot vision-based task and motion planning for ev battery disassembly and sorting
- Stolkin, R., Molitor, N., Berben, P., Verbeek, J., Reedman, T., Burtin, H., et al. (2023). Status, barriers and cost-benefits of robotic and remote systems applications in nuclear decommissioning and radioactive waste management. NEA report code: NEA/RWM/R(2022)1 in *White paper* (Paris, France: OECD Nuclear Energy Agency NEA), 168. Available online at: https://www.oecd-nea.org/upload/docs/application/pdf/2023-01/4b_-_nea_rwm_r_2022_1.pdf (Accessed December 19, 2025).
- Stolkin, R., Berben, P., Molitor, N., Reedman, T., and Burtin, H. (2024). Technoeconomic assessment of electric vehicle battery disassembly—challenges and opportunities from a robotics perspective. *IEEE Access*. doi:10.1109/ACCESS.2024.3520414
- Şucan, I. A., Moll, M., and Kavragi, L. E. (2012). The open motion planning library. *IEEE Robotics and Automation Mag.* 19, 72–82. doi:10.1109/MRA.2012.2205651
- Talha, M., Ghalamzan, E. A. M., Takahashi, C., Kuo, J., Ingamells, W., and Stolkin, R. (2016). "Towards robotic decommissioning of legacy nuclear plant: results of human-factors experiments with tele-robotic manipulation, and a discussion of challenges and approaches for decommissioning," in *2016 IEEE international symposium on safety, security, and rescue robotics (SSRR)* (IEEE), 166–173. doi:10.1109/SSRR.2016.7784294
- Touzani, H., Séguy, N., Hadj-Abdelkader, H., Suárez, R., Rosell, J., Palomo-Avellaneda, L., et al. (2022). Efficient industrial solution for robotic task sequencing problem with mutual collision avoidance and cycle time optimization. *IEEE Robotics Automation Lett.* 7, 2597–2604. doi:10.1109/LRA.2022.3142919
- Tsagkournis, E., Panagopoulos, D., Petousakis, G., Nikolaou, G., Stolkin, R., and Chiou, M. (2023). A supervised machine learning approach to operator intent recognition for teleoperated mobile robot navigation. *IFAC-PapersOnLine* 56, 8333–8338. doi:10.1016/j.ifacol.2023.10.1023
- Vitanov, I., Farkhatdinov, I., Denoun, B., Palermo, F., Otaran, A., Brown, J., et al. (2021). A suite of robotic solutions for nuclear waste decommissioning. *Robotics* 10, 112. doi:10.3390/robotics10040112
- Wang, H., and Chen, W. (2024). Task scheduling for heterogeneous agents pickup and delivery using recurrent open shop scheduling models. *Robotics Aut. Syst.* 172, 104604. doi:10.1016/j.robot.2023.104604
- Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., et al. (2022). "Finetuned language models are zero-shot learners," in *International conference on learning representations (ICLR)*.
- Yuan, Z., Wang, R., Kim, T., Zhao, D., Obi, I., and Min, B.-C. (2025). "Adaptive task allocation in multi-human multi-robot teams under team heterogeneity and dynamic information uncertainty," in *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (Atlanta, USA: IEEE).
- Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., et al. (2023). Instruction-following evaluation for large language models