Intent-Driven LLM Ensemble Planning for Flexible Multi-Robot Disassembly: Demonstration on EV Batteries

Cansu Erdogan[†,1] , Cesar Alan Contreras[†,1] , Alireza Rastegarpanah[†,1,2,*] , Manolis Chiou[3] , Rustam Stolkin[1]

[1]Extreme Robotics Lab, School of Metallurgy & Materials, University of Birmingham, Birmingham, UK

[2]Department of Applied Artificial Intelligence and Robotics, School of Computer Science, Aston University, Birmingham, UK [3]School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

Abstract—This paper addresses the problem of planning complex manipulation tasks, in which multiple robots with different end-effectors and capabilities, informed by computer vision, must plan and execute concatenated sequences of actions on a variety of objects that can appear in arbitrary positions and configurations in unstructured scenes. We propose an intent-driven planning pipeline which can robustly construct such action sequences with varying degrees of supervisory input from a human using simple language instructions. The pipeline integrates: (i) perception-to-text scene encoding, (ii) an ensemble of large language models (LLMs) that generate candidate removal 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 battery for recycling applications. A variety of components must be grasped and removed in specific sequences, determined by human instructions and/or by task-order feasibility decisions made by the autonomous system. 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 with human participant experiments. Results indicate that our ensemble-with-verification approach reliably maps operator intent to safe, executable multi-robot plans while maintaining low user effort.

Index Terms—Large Language Models, Multi-Robot Disassembly, Task Planning, Intent Recognition, Human-Robot Interaction

I. INTRODUCTION

Traditional pre-programmed automation has worked well in highly structured settings, becoming widespread in automotive manufacturing and other factory assembly tasks since the 1970s. However, unstructured, variable, and uncertain environments pose additional challenges which are a bottle-neck to robotization of many other societally important industries. The disassembly of electric-vehicle (EV) batteries is a prototypical task that demands strong generalisation from multi-robot systems. Battery packs vary widely across makes, models, and model years; there is no stable standard, and designs evolve continuously, so the classic "re-program on every new variant" approach is untenable [1]. Instead, scalable automation must (i) use computer vision to robustly recognise and localise heterogeneous components and infer how they fit together, (ii) plan low-level actions (grasping, moving, unscrewing, 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 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; 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 [2], motivating the intent-driven, perception-grounded approach we pursue in this paper.

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 synthesise contextual informa-

tion, 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 modelling effort and enables a more adaptive, scalable planning pipeline.

A. Overview

This study introduces a multi-LLM-based task Planning framework for multi-robot disassembly operations. A user writes what they want 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 and robot platforms with heterogeneous capabilities; (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, providing a simplified hand-off from plan to execution to reduce operator cognitive and interaction workload.
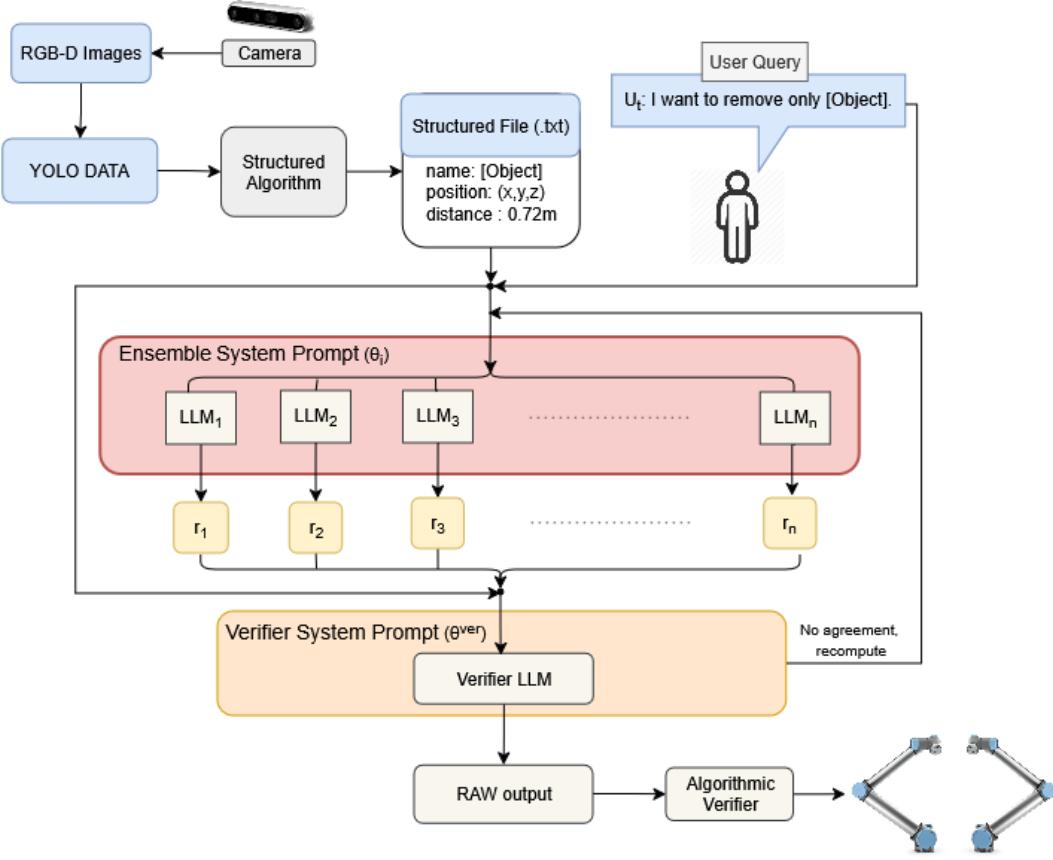

Fig. 1: A task-planning pipeline that enhances reliability and flexibility by using an ensemble of LLMs (same checkpoint, different seeds) to generate diverse solutions from structured inputs. These candidate plans are then synthesised and validated by a dedicated verification module before robot execution.

II. Literature Review

Our setup comprises multiple components, including task planning, large language models, ensemble methods, operator intent, and adaptive autonomy. This section reviews the relevant work for each component and motivates the design choices used later.

A. 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 for ensuring efficient and collision-free movements in shared environments. Classical optimisation methods are often used, such as Mixed-Integer Linear Programming (MILP) and Constraint Programming (CP), that encode task assignment under resource and precedence constraints [3], [4]. These approaches are very effective in structured industrial scenarios. Still, they may suffer from scalability issues and degrade their performance in dynamic or uncertain environments.

To improve scalability and adaptation, recent works have explored the use of heuristic and distributed scheduling methods [5], [6], which show more responsive and decentralised decision-making capabilities under uncertainties. Knowledge-based and semantic reasoning frameworks [7] 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 modelling.

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.

B. 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. 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 [8]. However, these benchmarks typically represent static or highly structured environments. Real-world multi-robot systems tend to operate in 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 VLMs for object localisation. This integration improves generalisation in robotic manipulation [9]. Another framework working in this direction is ManipLLM, which introduces a multimodal LLM-based manipulation system that focuses on object-centric reasoning [10]. Unlike earlier learning-based methods limited to predefined object categories, ManipLLM employs commonsense reasoning chains and test-time adaptation strategies to generalise across diverse manipulation tasks, performing well in simulated and real-world environments.

For human–robot interaction (HRI), recent work has emphasised collaboration with LLMs, including studies of LLM-enabled robots performing execution, negotiation, selection, and plan generation [11], as well as vision-language-action reasoning combined with teleoperation [12]. Findings suggest that while LLMs excel in affective and negotiation-based interactions, they struggle to maintain logical consistency and to support creative collaboration.

In our work we operationalise 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.

C. 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 [13], and sometimes is followed by a reinforcement learning step with human feedback (RLHF) using their preferences data [14]. Exposing the systems to very diverse instructional data and human feedback gives strong zero-shot generalisation on unseen tasks, and also helps reduce toxic behaviours [13], [14], [15]. 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 [13] 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 [15].

The increase of IT models prompted the design of specialised benchmarks that evaluate models on their obedience to explicit constraints, with some of these including: **IFEval**, which judges whether generated text satisfies verifiable instructions (e.g length or lexical constraints) [16], **FollowBench** dissects adherence across content, format, and style dimensions [17] and **M-IFEval**, which adds judgement of multilingual instructions, comparing it to the default IFEval [18].

Ensemble methods can complement Instruction Tuning. Deep ensembles improve calibration by aggregating independently sampled models [19]. Model-library selection ensembles pick a diverse subset from thousands of candidates for optimal accuracy [20]. Sparse mixture-of-expert (MoE) architectures, using the Switch Transformer, scale to trillions of parameters and some obtain ensemble-like gains fusing final layers at fixed compute by routing a subset of tokens through selected layers [21], [22]. Ensembles of instruction-tuned generators in theory could produce higher-quality synthetic training data, and 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 are still limited.

In our work, instead of extra fine-tuning, we perform 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.

D. 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 motions showed that predicting a collaborator's next goal allows a robot to adapt its plan, yielding improvements in both objective and perceived performance [23]. 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 [24], [25], [26] reaching real-time accuracy comparable to hand-engineered baselines.

Vision-only, zero-shot intent recognition has also shown reductions in operator effort on unseen manipulation tasks [27]. 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 [28]. 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 [29].

Taking from the literature the importance of assistance and intent recognition, in this work we treat natural-language prompts as the explicit operator intent and fuse them with the scene text so the ensemble orders only disassembly orders 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, giving this calculations to the system keeps cognitive-load low, while it also preserves trust by allowing operators to verify and change disassembly order before execution.

E. 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 on-board autonomy in response to context, workload, mission phase, or robot conditions [30], [31], [32], [33]. Some adjustable-autonomy frameworks formalise transfer of control strategies, balancing decision quality against miscoordination costs, and show performance gains for single operators supervising several robots at varying autonomy levels [34], [35]. More recently, some research has framed VA as a prerequisite for trustworthy AI, emphasising transparency and explainability as determinants of when autonomy should be ceded or reclaimed [31].

Language-centric VA mechanisms in multi-robot tasks on VR testbeds have also been worked on [36], 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 to do Variable Autonomy. Variable autonomy rules and allocation rules can also be learned, as an example, the ATA-HRL framework utilises 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 [37].

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, allowing to keep a low cognitive workload, and the system also lets operators take control at any time.

III. METHOD

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 test our dynamically adaptive method, we build on our previous papers [38], [39], and have designed a robotic scene consisting of two UR10e robots with cameras mounted on their end effectors, with the task of identifying objects forming part of a dissassembly task (Fig. 2), and correctly disassembling them. For this task, we used weights of YOLOv8 with 226 images, similarly to our previous paper [39] components, in a real-world environment. In addition to the usage of the pre-trained Qwen-32B LLM, without any additional fine-tuning.
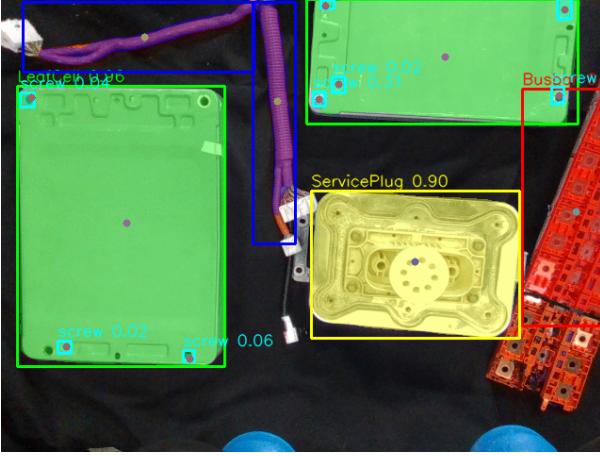

Fig. 2: Example perception output used by the LLM-ensemble planner. A YOLOv8 model overlays class labels and confidence scores for the battery components (*Leafcell*, *Cable*, *Busbar*, *Service Plug*, *Screw*).

We deployed our experimental setup across two dedicated desktop computers. The first system, denoted as PC_1, was responsible for real-time perception and robotic control. PC_1 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 (Fig. 3), which continuously provided synchronised RGB and depth data. Additionally, PC_1 managed low-level control of two Universal Robots (UR-series) manipulators, handling joint commands and execution of planned motions.

The second system, designated PC_2, performed high-level reasoning tasks and hosted the language models utilised in our experiments. PC_2 was configured with an Intel i9 CPU, an NVIDIA RTX 5090 GPU, and 32 GB of RAM. This computer exclusively ran a quantised (4-bit) version of the Qwen-32B large language model. All models were served through an Ollama-compatible API, and inter-system communication between PC_1 and PC_2 occurred over a WiFi network, with 200ms latency.

A. 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 serialised $\sigma_t = \text{Serialise}(s_t)$.

Excerpt of Unstructured Text s_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 ...
```

Excerpt of Structured Text σ_t

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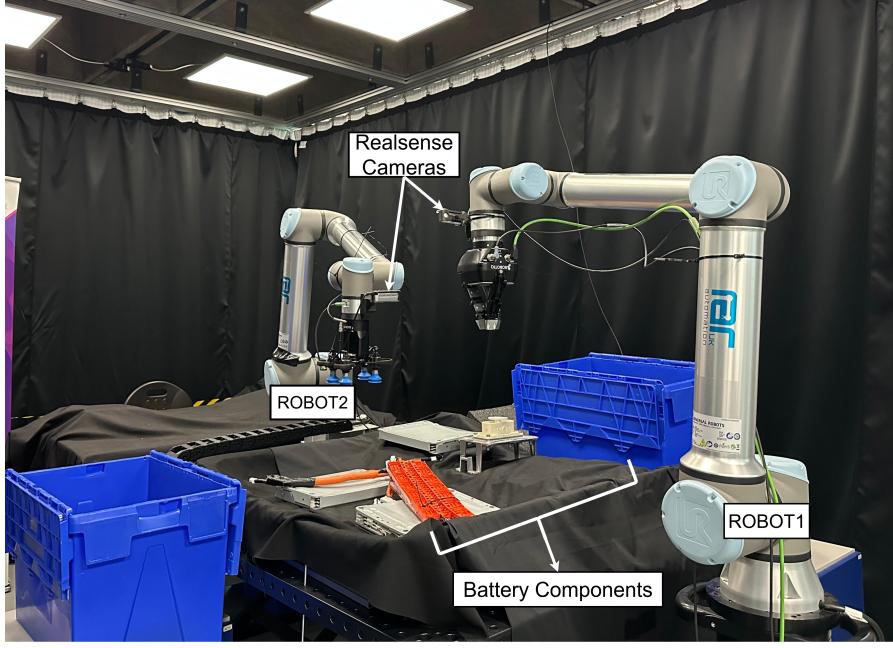
Fig. 3: Data-collection setup: two UR10e collaborative robot arms (ROBOT1 and ROBOT2) equipped with Intel RealSense RGB-D cameras and the components arranged on the workbench.

B. Coordinate rewrite

Every Cartesian triple $[x, y, z]$ inside σ_t is replaced by $[-z, x, y, d]$ with

$$d = \sqrt{x^2 + y^2 + z^2}, \tag{1}$$

, due to the transformations from the camera to spatial positions on the operator frame, while all other tokens are left untouched. Write

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

for the transformed scene text produced by the coordinate rewriting.

C. 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, 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-actions section forbids headings, blank lines, fabricated data, or any explanatory text. Finally, a bank of illustrative examples anchors the model's behaviour 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
```

D. Generator pass

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

---

Final User Message U_t

```
Here is the data to use: ⟨D_t⟩
This is what I want to do 'extra prompt':
⟨u_t^extra⟩
/no_think
```

With sampling seed s_j, the message sent to the LLM API is

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

Generator LLM Message

```
"system", θᵢ,
"user", Uₜ.
```

E. LLM candidate generation

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

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

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

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

F. 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 fulfils 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 θ^{ver}

```
- Role and responsibilities definition
- Structured input description
- Generalised algorithm recap
- Validation criteria
- Decision and selection logic
- Explicit output formatting rules
- Restricted actions and prohibitions
```

G. 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. Denote by $\mathcal{R}_t = \{\, r_{j,t} \mid s_j \in \mathcal{S} \,\}$ The set of seed-specific candidate lists harvested in (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.

Verifier LLM Message

```
system: θᵛᵉʳ
user:
=== ORIGINAL CONTEXT ===
⟨Cₜ⟩

=== CANDIDATE RESPONSES ===
Response 1:
⟨r₁,ₜ⟩
Response 2:
⟨r₂,ₜ⟩
⋮
Response |S|:
⟨r|S|,ₜ⟩
/no_think
```

H. 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 $\hat{\mathbf{q}}^{(n)} = (L, U, Z)^{\mathsf{T}}$. The object

coordinates are then mapped back to the original frame via $(L, U, Z) \mapsto (x, y, z) = (U, Z, -L)$, yielding

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

I. Algorithmic deterministic verifier/filter

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

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

The accepted list is therefore

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

J. Robot action execution

The high-level object lists generated by the LLM-ensemble are executed by a single `robot_control` ROS node that commands both UR10e arms (ROBOT1 & ROBOT2). Built on MoveIt, this node performs motion planning, trajectory streaming, and run-time safety checks in one process, thereby eliminating the latency and synchronisation issues that arise when an independent planner drives each arm.

Control pipeline,

- *Scene initialisation* - 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 [40], velocity limits: 20% of maximum.

For further implementation details, please refer to [39].

K. 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.

IV. Experiment

To utilise 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 that the framework, with the selected LLM, generalises across a wide variety of conditions, contexts, and instructions. The goal for 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 [41]. The collected dataset consisted of 200 RGB-D images, obtained with a RealSense 435i stereo camera. All RGB images passed through our YOLOv8 weights, which provided five different classes (screw, leafcell, cable, busbar and service plug)

Figure 3 and Figure 4 show an example of object distribution and robot placement in the scene. Additionally, we analysed the object placement distribution by stacking every RGB frame of all trials, creating a blend of the 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 surface infrequent positions of components that would otherwise vanish in a simple average.

A. Selection of Language Model

Reliable plan generation demands an LLM that (i) follows our specialised instruction-example prompt format without 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
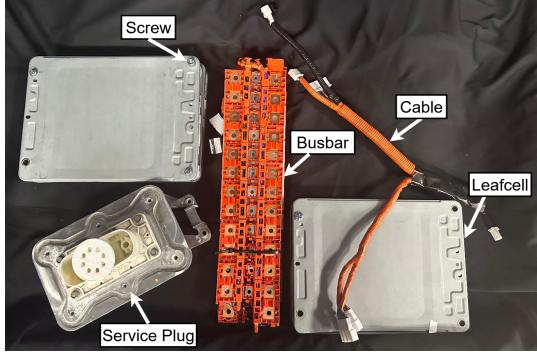
Fig. 4: Sample scene showing five battery components (Leafcell, Busbar, Cable, Service Plug and Screw)

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

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 I).

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

The model **qwen3:32b** completed all five trials, though with a moderate median latency of 21s. 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 appear in Table I.

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

B. 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**.

TABLE I: 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 21s (range 18-24s). 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, giving unnecessary explanations, even though runtimes stayed within 30-46s. Errors stem from systematic instruction following issues rather than timeouts.
phi4-reasoning:latest	5	No valid output across all runs. Latency was highly unstable (16s-144s), indicating internal difficulties in reasoning steps, and overthinking of instructions.
phi4-reasoning:plus	3	Succeeded in the first two trials (approximately 39-48s) but failed 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 127-129s). No successful completions observed.
gemma3:27b-it-qat	3	Two successful runs (latencies 33s and 70s) 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 180s cutoff in all five trials; treated as hard failures with no usable output.
qwen3:30b-a3b	3	Completed two trials (in approximately 36s) but failed three. Failures produced verbose, off-specification answers in under 10s.

The single-model baseline consisted solely of the primary model (qwen3:32b), which 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.
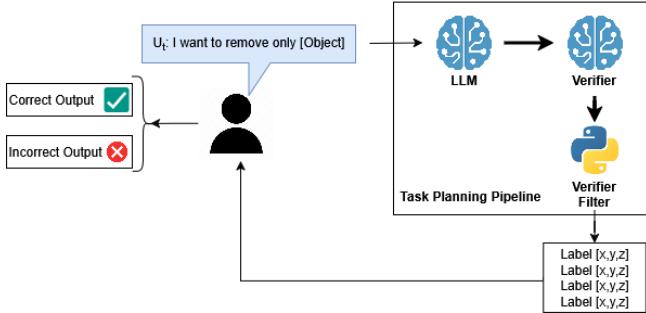

Fig. 6: LLM + verifier architecture distinguishing correct (✓) and incorrect (×) output for full evaluation

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, 9000 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.

C. 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 summarised 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 9000 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.
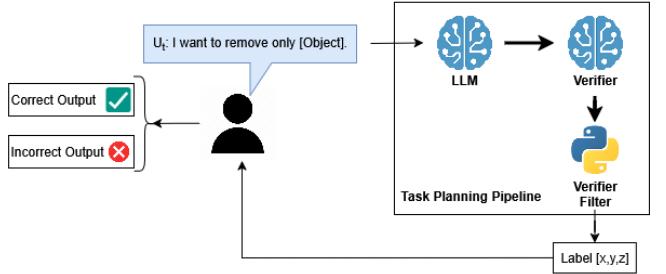

Fig. 7: LLM + verifier architecture distinguishing correct (✓) and incorrect (×) output for the first object

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.

D. 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 give 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 saw two example intents during training (e.g., "remove only the leafcell from the left module"), then wrote their own intents for each scene. The study 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 300s baseline of an expert operator performing the task without LLM assistance is recorded.

Participants reviewed each object appearing on 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: (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; (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: (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 contextualise 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.

V. 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 at various configurations is conducted, comparing the different ablations of the system. Third, the workload analysis and real-world assessment results.

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.

A. 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 summarises 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*, *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.

B. Pipeline Results

We evaluated the pipeline on an operator-prepared fixed dataset of scenes and prompts, separate 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 just 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 with 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 rest of the list. Timing uses the same executions for both analyses. Therefore, average and median times are reported once per configuration and shared across the Full and Next evaluations.

On Table II, the accuracies for the different configurations of *1 LLM*, *3 LLMs* ensemble + verifier (Ensemble), and ensemble + verifier + algorithmic verifier (Final), and *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

TABLE II: 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

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 for observation of the distribution of time per request across the dataset, with Figure 11 complementing this, by reporting the distribution of final stage times for the 9000 runs with density values for each of
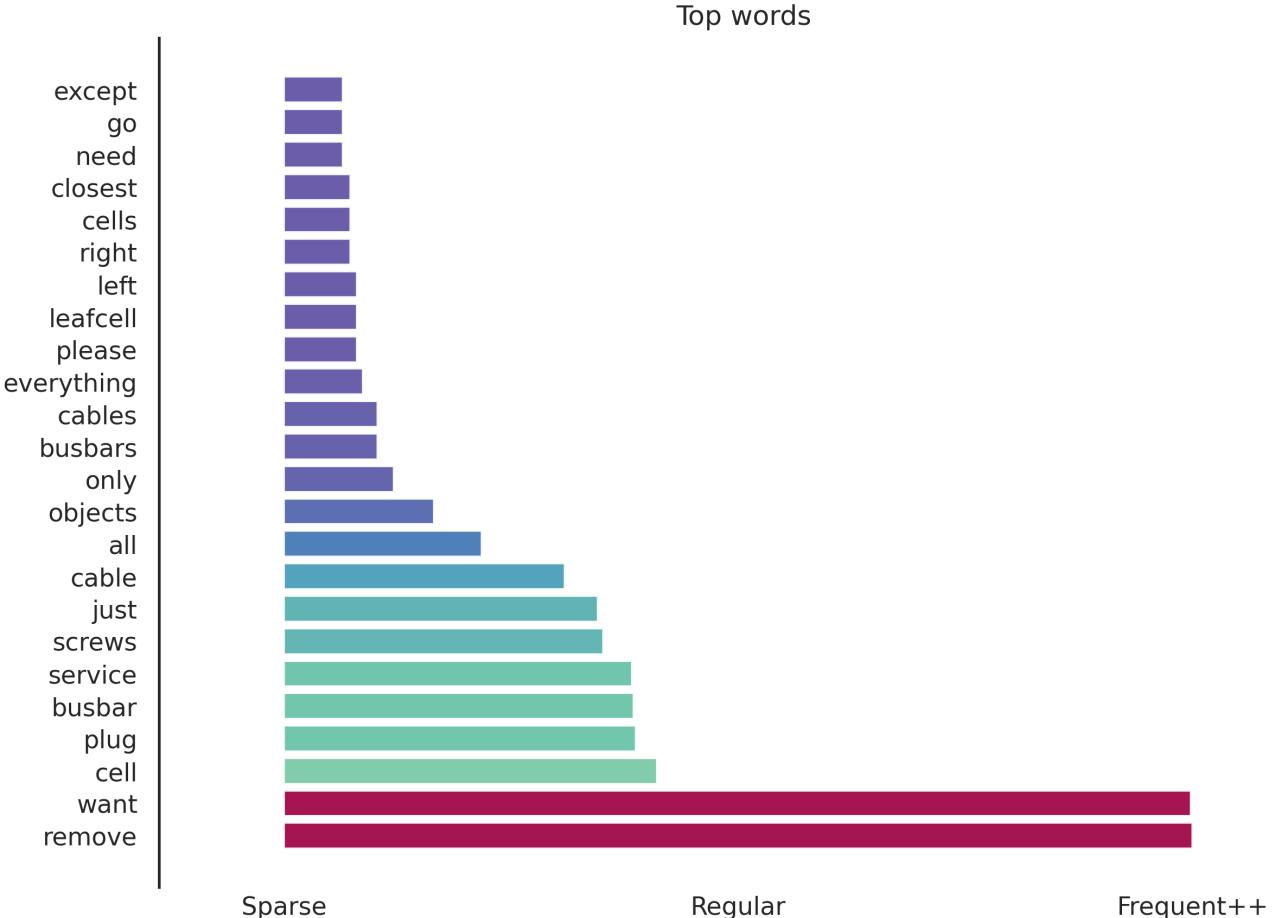

Fig. 8: Token frequency across 600 operator requests. Bars are ordered from rare to common. Colours grade from purple (sparse) through teal to red (frequent). Red bars highlight the two dominant verbs (*want*, *remove*); the teal gradient marks object classes (*LeafCell*, *Service Plug*, *Busbar*, *Screw*, *Cable*) following in prominence. At the same time, other less prominent words show in purple.
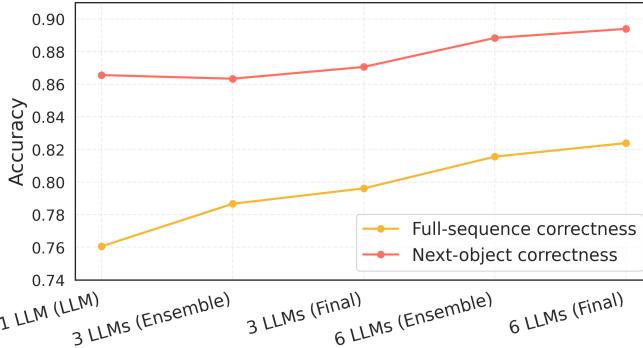
Fig. 9: Accuracy across pipeline configurations for Full and Next-object evaluations.
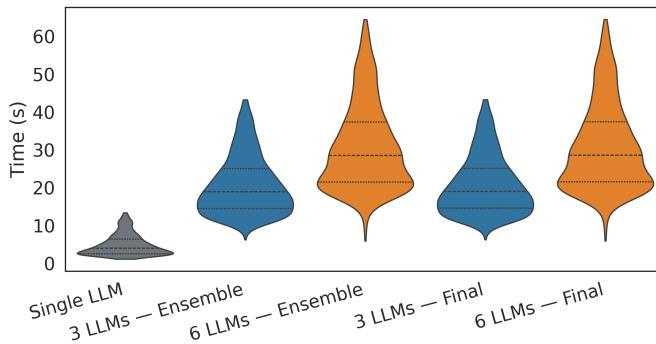
the time bandwidths.

Fig. 10: End-to-end per-request runtime cleaned Interquartile range distribution with embedded boxplot.
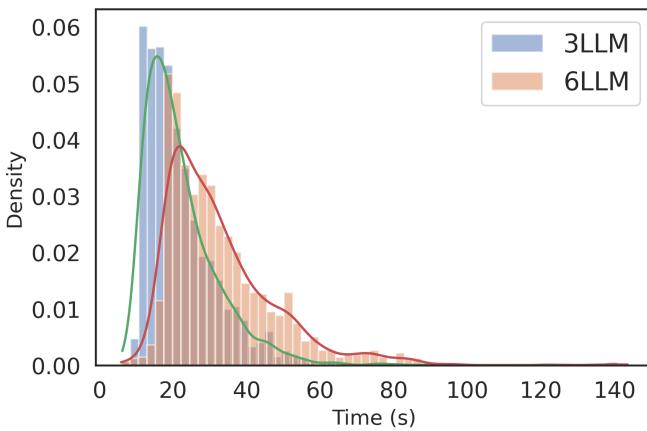

Fig. 11: Final-stage runtime distribution (9,000 runs). Density-normalised histograms with KDE for 3LLM and 6LLM pipelines; applies to both Full and Next object evaluations. Effective bandwidths are ≈2.45 s (3 LLM) and ≈3.36 s (6 LLM).

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, Final). Within each stage,

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

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 is a single request from the same executions as Table II. The model tokeniser measures tokens; latency is the time recorded for the Final stage. The separate series indicate the 3LLM and 6LLM ensembles.

Table III lists p-values for pairwise stage comparisons (LLM vs Ensemble, LLM vs Final, Ensemble vs Final) under the Full (all objects) and Next (next object) settings, for 3LLM and 6LLM configurations. Entries with p 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 (1800). 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 operator intent under our scoring rules.

C. 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), totalling 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 IV reports the mean and median completion time for the participants using the pipeline for the control of the robots. A 300s manual baseline serves as a contextual reference from an expert operator, performing the task without LLM-assisted planning, with which to compare this data.

Table V summarises the global NASA-TLX workload (0-100). Used to report the overall perceived workload for the session.

Table VI reports mean ± SD and median for each NASA-TLX subscale (0-100): Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, and Frustration.
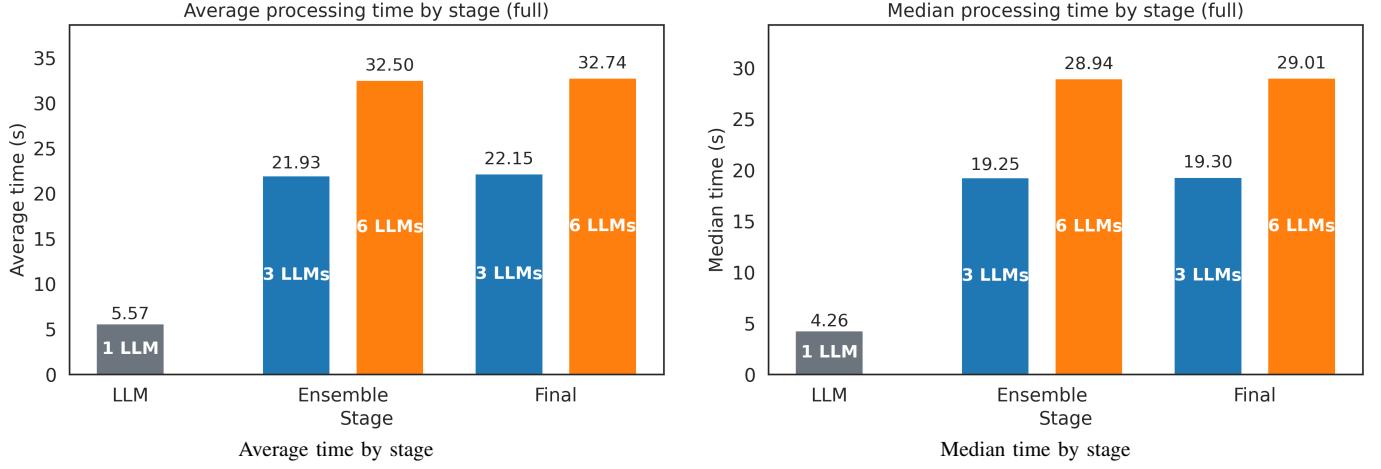
Fig. 12: Processing time by stage. Left: average time; right: median time. The x-axis lists stages (*LLM*, *Ensemble*, *Final*). Within each stage, bars compare configurations: single-model for the 1 LLM configuration, and 3-model vs 6-model for *Ensemble* and *Final*. The y-axis is seconds.
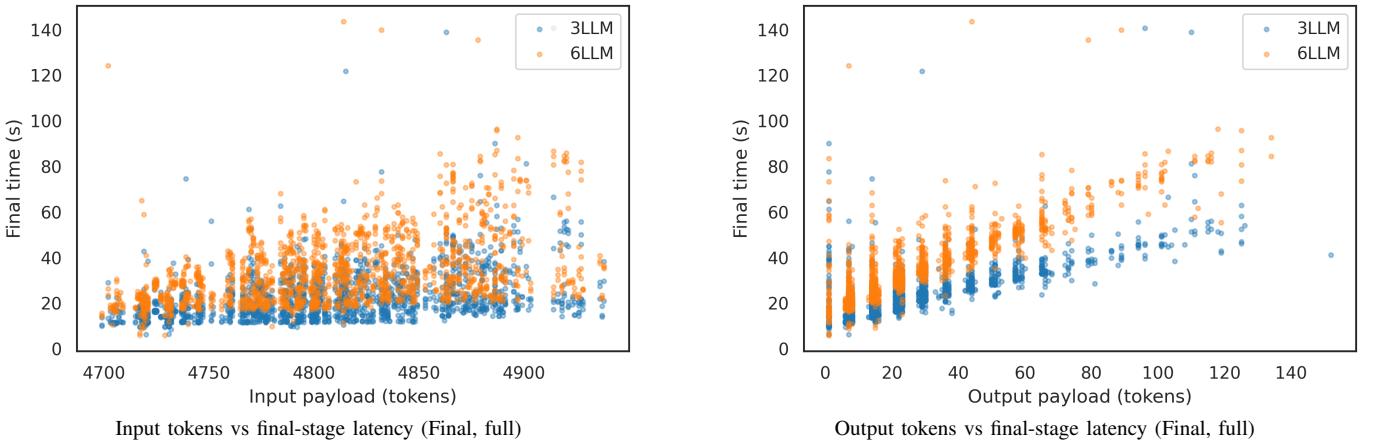

Fig. 13: Payload size vs latency until end of the *Final* ensemble stage. Left: input token count; right: output token count. Each point is a single request from the same executions as Table II. Tokens are counted by the model tokeniser; latency is the time of the *Final* stage. Separate series are shown for the 3-model and 6-model ensembles.

TABLE III: 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}	✓

TABLE IV: Completion time (pipeline condition). Values are Mean $\pm$ SD and Median, in seconds.

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

TABLE V: Global NASA-TLX workload (0-100). Values are Mean $\pm$ SD and Median.

Metric	Mean $\pm$ SD	Median
NASA-TLX (Global)	14.95 ± 11.80	9.67

.

TABLE VI: NASA-TLX subscales (0-100): Mean $\pm$ SD and Median across participants.

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

VI. DISCUSSION

This work tested intent-driven ensemble LLM planners, performing a case study on EV-battery component disassembly using two UR10e arms. However, it's capable of adapting to different environments, robots, and scenarios.

A. Model selection

As the model selection and preliminary experiments show, the most important characteristic when choosing among low parameter count models was their instruction adherence capability. The most common errors found in models that didn't 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 seconds on average.

B. Pipeline correctness

Across the 9000 runs (Table II), **full object** correctness rose 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 3LLM and 0.8 pp for 6LLM) while removing hallucinated objects at nearly 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: 3LLM $p = 2.19 \times 10^{-7}$, 6LLM $p = 1.57 \times 10^{-9}$; LLM $\rightarrow$ Final: 3LLM $p = 6.61 \times 10^{-10}$, 6LLM $p = 1.19 \times 10^{-11}$). Ensemble $\rightarrow$ Final yields small but significant gains (Full: 3LLM $p = 3.61 \times 10^{-5}$, 6LLM $p = 1.74 \times 10^{-3}$); see Table III.

For **next-object** correctness, the single model gives a high accuracy (0.866). The 3LLM configuration achieves 0.863 (Ensemble) and 0.871 (Final), while the 6LLM configuration reaches 0.888 (Ensemble) and 0.894 (Final), indicating an increasing trend with a slight dip at 3LLM before rising at 6LLM. Paired t-tests show significant gains over the single model (3LLM: LLM $\rightarrow$ Ensemble $p = 2.90 \times 10^{-2}$, LLM $\rightarrow$ Final $p = 1.45 \times 10^{-3}$; 6LLM: $p = 1.13 \times 10^{-3}$ and 7.70×10^{-5}). Ensemble $\rightarrow$ Final adds further small but significant improvements (3LLM $p = 7.78 \times 10^{-4}$; 6LLM $p = 2.68 \times 10^{-3}$); Table III.

Latency increases with ensemble size: mean end-to-end time is 5.57s (single model), 21.93–22.15s (3-model, ±filter), and 32.50–32.74s (6-model, ±filter) (Table II). Median times show the same pattern (4.26s $\rightarrow$ 19.25–19.30s $\rightarrow$ 28.94–29.01s). The final filter adds approximately 0.2 seconds on average, which is negligible compared to ensemble sampling and verification. Distributions are right-tailed (Fig. 10) and widen with ensemble size, with the final-stage histograms (Fig. 11) showing the bandwidth increasing from $\approx$2.45s (3-model) to $\approx$3.36s (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 compared to 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 Figures 13 suggest 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 (Fig. 12), acting as a very effective low-cost last line of defence against hallucinations. For our framework the scene included five component classes (leafcell, busbar, cable, service plug, screw, Figure 4) detected by YOLOv8, although the LLM model was never given the explicit labels it was able to properly adapt with the provided instructions, to the components of the disassembly task, making a strong case for the generalizability of the pipeline, which cared more about the data format and structure consistency (with no model working in preliminary tests, with unstructured data).

The bar plot of prompt tokens (Fig. 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.

C. 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, perceived workload was low [42], [43](NASA–TLX mean 14.95, median 9.67) after a brief single training session (Tables IV–VI). 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 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; the operator authorises execution.

VII. 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 rises from 0.761 to 0.824 with a 6LLM ensemble plus an LLM verifier and a final deterministic filter; next-object accuracy rises 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 (300s). Time-to-action was faster

than the manual baseline by roughly one-third on average ($\approx 34\%$ faster), with the median about 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.

A. Limitations and Future Work

Our ensemble improves plan correctness but shows the expected accuracy–latency trade-off: accuracy gains taper while 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, as well as input-complexity gating between 1/3/6 models. We also plan to cache and reuse verified partial plans for recurring intents (when possible).

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, graspability) to reject impossible steps and to provide minimal counter examples 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 test other assembly/disassembly settings (e-waste, other types of end-of-life product dismantling, nuclear decommissioning). Another area for deeper study in future work, is expanding the human user study to include larger numbers of participants, and exploring if and how human-perceived workload varies across different types of tasks and environments.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

Author Contributions

Conceptualisation, C.E. and C.A.C.; methodology, C.E. and C.A.C.; software, C.E. and C.A.C.; validation, C.E., C.A.C. and M.C.; investigation, C.E., C.A.C. and A.R.; data curation, C.E. and C.A.C.; writing—original draft preparation, C.E., C.A.C; writing—review and editing, C.E., C.A.C., R.S., M.C. and A.R.; visualisation, C.E. and C.A.C; supervision, A.R. and R.S.; project administration, A.R., R.S.; funding acquisition, A.R. and R.S. All authors have read and agreed to the published version of the manuscript.

References

[1] J. Hathaway, C. A. Contreras, M. E. Asif, R. Stolkin, and A. Rastegarpanah, "Technoeconomic assessment of electric vehicle battery disassembly–challenges and opportunities from a robotics perspective," *IEEE Access*, 2024.

[2] M. E. Asif, A. Rastegarpanah, and R. Stolkin, "Robotic disassembly for end-of-life products focusing on task and motion planning: A comprehensive survey," *Journal of Manufacturing Systems*, vol. 77, pp. 483–524, 2024.

[3] S. Fatemi-Anaraki, R. Tavakkoli-Moghaddam, M. Foumani, and B. Vahedi-Nouri, "Scheduling of multi-robot job shop systems in dynamic environments: Mixed-integer linear programming and constraint programming approaches," *Omega*, vol. 115, p. 102770, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305048322001773

[4] H. Touzani, N. Séguy, H. Hadj-Abdelkader, R. Suárez, J. Rosell, L. Palomo-Avellaneda, and S. Bouchafa, "Efficient industrial solution for robotic task sequencing problem with mutual collision avoidance & cycle time optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2597–2604, 2022.

[5] H. Wang and W. Chen, "Task scheduling for heterogeneous agents pickup and delivery using recurrent open shop scheduling models," *Robotics and Autonomous Systems*, vol. 172, p. 104604, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889023002439

[6] T. Pan, C. K. Verginis, and L. E. Kavraki, "Robust and safe task-driven planning and navigation for heterogeneous multi-robot teams with uncertain dynamics," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 3482–3489.

[7] R. Bernardo, J. M. Sousa, and P. J. Gonçalves, "A novel framework to improve motion planning of robotic systems through semantic knowledge-based reasoning," *Computers & Industrial Engineering*, vol. 182, p. 109345, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835223003698

[8] Z. Long, G. Killick, R. McCreadie, and G. A. Camarasa, "Robollm: Robotic vision tasks grounded on multimodal large language models," 2024. [Online]. Available: https://arxiv.org/abs/2310.10221

[9] R. Garcia, S. Chen, and C. Schmid, "Towards generalizable vision-language robotic manipulation: A benchmark and llm-guided 3d policy," 2025. [Online]. Available: https://arxiv.org/abs/2410.01345

[10] X. Li, M. Zhang, Y. Geng, H. Geng, Y. Long, Y. Shen, R. Zhang, J. Liu, and H. Dong, "Manipllm: 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, 2024. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024/papers/Li_ManipLLM_Embodied_Multimodal_Large_Language_Model_for_Object-Centric_Robotic_Manipulation_CVPR_2024_paper.pdf

[11] C. Kim, C. Lee, and B. Mutlu, "Understanding large-language model (llm)-powered human-robot interaction," 03 2024, pp. 371–380.

[12] H. Liu, Y. Zhu, K. Kato, A. Tsukahara, I. Kondo, T. Aoyama, and Y. Hasegawa, "Enhancing the llm-based robot manipulation through human-robot collaboration," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, p. 6904–6911, aug 2024. [Online]. Available: http://dx.doi.org/10.1109/LRA.2024.3415931

[13] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," in *International Conference on Learning Representations (ICLR)*, 2022. [Online]. Available: https://openreview.net/forum?id=gEZrGCozdqR

[14] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html

[15] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, "Scaling instruction-finetuned language models," *Journal of Machine Learning Research*, vol. 25, pp. 1–53, 2024. [Online]. Available: http://jmlr.org/papers/v25/23-0870.html

[16] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou, "Instruction-following evaluation for large language models," 2023. [Online]. Available: https://arxiv.org/abs/2311.07911

[17] Y. Jiang, Y. Wang, X. Zeng, W. Zhong, L. Li, F. Mi, L. Shang, X. Jiang, Q. Liu, and W. Wang, "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)*. Association for Computational Linguistics, 2024. [Online]. Available: https://aclanthology.org/2024.acl-long.257/

[18] A. Dussolle, A. C. Díaz, S. Sato, and P. Devine, "M-ifeval: Multilingual instruction-following evaluation," in *Findings of the Association for Computational Linguistics: NAACL 2025*. Albuquerque, USA: Association for Computational Linguistics, 2025. [Online]. Available: https://aclanthology.org/2025.findings-naacl.344/

[19] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2017. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html

[20] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 18.

[21] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, 2022. [Online]. Available: https://www.jmlr.org/papers/volume23/21-0998/21-0998.pdf

[22] D. Raposo, S. Ritter, B. Richards, T. Lillicrap, P. C. Humphreys, and A. Santoro, "Mixture-of-depths: Dynamically allocating compute in transformer-based language models," 2024. [Online]. Available: https://arxiv.org/abs/2404.02258

[23] C. Liu, J. B. Hamrick, J. F. Fisac, A. D. Dragan, J. K. Hedrick, S. S. Sastry, and T. L. Griffiths, "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)*. International Foundation for Autonomous Agents and Multiagent Systems, 2016. [Online]. Available: https://arxiv.org/abs/1802.01780

[24] E. Tsagkournis, D. Panagopoulos, G. Petousakis, G. Nikolaou, R. Stolkin, and M. Chiou, "A supervised machine learning approach to operator intent recognition for teleoperated mobile robot navigation," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 8333–8338, 2023, 22nd IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896323014064

[25] C. A. Contreras, M. Chiou, A. Rastegarpanah, M. Szulik, and R. Stolkin, "Probabilistic human intent prediction for mobile manipulation: An evaluation with human-inspired constraints," *arXiv preprint arXiv:2507.10131*, 2025.

[26] D. Panagopoulos, G. Petousakis, R. Stolkin, G. Nikolaou, and M. Chiou, "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, 2021, pp. 125–131.

[27] A. Belsare, Z. Karimi, C. Mattson, and D. S. Brown, "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, 2025. [Online]. Available: https://dl.acm.org/doi/10.5555/3721488.3721519

[28] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2016.

[29] M. Bowman, J. Zhang, and X. Zhang, "Intent-based task-oriented shared control for intuitive telemanipulation," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 4, dec 2024. [Online]. Available: http://dx.doi.org/10.1007/s10846-024-02185-1

[30] C. A. Contreras, A. Rastegarpanah, M. Chiou, and R. Stolkin, "A mini-review on mobile manipulators with variable autonomy," *Frontiers in Robotics and AI*, vol. 12, p. 1540476, 2025.

[31] L. Methnani, M. Chiou, V. Dignum, and A. Theodorou, "Who's in charge here? a survey on trustworthy ai in variable autonomy robotic systems," *ACM computing surveys*, vol. 56, no. 7, pp. 1–32, 2024.

[32] M. Chiou, N. Hawes, and R. Stolkin, "Mixed-initiative variable autonomy for remotely operated mobile robots," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 10, no. 4, pp. 1–34, 2021.

[33] T. Reinmund, P. Salvini, L. Kunze, M. Jirotka, and A. F. Winfield, "Variable autonomy through responsible robotics: Design guidelines and research agenda," *ACM Transactions on Human-Robot Interaction*, vol. 13, no. 1, pp. 1–36, 2024.

[34] P. Scerri, D. V. Pynadath, and M. Tambe, "Towards adjustable autonomy for the real world," *Journal of Artificial Intelligence Research*, vol. 17, pp. 171–228, 2002.

[35] J. W. Crandall and M. A. Goodrich, "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)*, vol. 3. IEEE, 2001, pp. 1624–1629.

[36] Y. Lakhnati, M. Pascher, and J. Gerken, "Exploring a gpt-based large language model for variable autonomy in a vr-based human-robot teaming simulation," *Frontiers in Robotics and AI*, vol. 11, apr 2024. [Online]. Available: http://dx.doi.org/10.3389/frobt.2024.1347538

[37] Z. Yuan, R. Wang, T. Kim, D. Zhao, I. Obi, and B.-C. Min, "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, 2025. [Online]. Available: https://arxiv.org/abs/2409.13824

[38] C. Erdogan, C. A. Contreras, R. Stolkin, and A. Rastegarpanah, "Multi-robot task planning for efficient battery disassembly in electric vehicles," *Robotics*, vol. 13, no. 5, 2024. [Online]. Available: https://www.mdpi.com/2218-6581/13/5/75

[39] A. Shaarawy, C. Erdogan, R. Stolkin, and A. Rastegarpanah, "Multi-robot vision-based task and motion planning for ev battery disassembly and sorting," 2025. [Online]. Available: https://arxiv.org/abs/2509.21020

[40] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[41] F. Jacob Solawetz, "What is yolov8? the ultimate guide," https://blog.roboflow.com/whats-new-in-yolov8/, 2025, accessed:2 July 2025.

[42] C. Nikulin, G. Lopez, E. Piñonez, L. Gonzalez, and P. Zapata, "Nasa-tlx for predictability and measurability of instructional design models: case study in design methods," *Educational Technology Research and Development*, vol. 67, no. 2, pp. 467–493, 2019.

[43] R. A. Grier, "How high is high? a meta-analysis of nasa-tlx global workload scores," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 59, no. 1. Sage Publications Sage CA: Los Angeles, CA, 2015, pp. 1727–1731.