Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina 2026

Arquitectura cognitiva híbrida para remanufactura: integrando abstracción semántica con planificación PDDL

Cabezas-Lopez, G.[a], Zalama-Casanova, E.[a,b], Gómez-García-Bermejo, J.[a,b]

[a] *Centro Tecnológico CARTIF, Parque Tecnológico de Boecillo, Av Francisco Vallés 4, 47151 Boecillo, España.*
[b] *Departamento de Ingeniería de Sistemas y Automática, Universidad de Valladolid, Escuela de Ingenierías Industriales, Paseo Prado de la Magdalena 3-5, 47011, Valladolid, España.*

Resumen

La transición hacia la Industria 5.0 demanda sistemas robóticos capaces de colaborar con humanos, no solo en líneas de producción estructuradas, sino también en procesos menos estandarizados de de recuperación de valor, en particular la remanufactura. Este contexto presenta desafíos únicos: alta variabilidad en los objetos entrantes y baja cadencia de producción, penalizando la programación robótica tradicional. Si bien los Grandes Modelos de Lenguaje (LLMs) ofrecen generalización semántica, su aplicación carece de la robustez necesaria para secuencias de desmontaje críticas. Este artículo propone una arquitectura híbrida y modular para la orquestación y planificación robótica en entornos de remanufactura. A diferencia de enfoques end-to-end, nuestra propuesta desacopla el razonamiento semántico de la planificación determinista. El sistema utiliza modelos de percepción y LLMs para abstraer el entorno no estructurado en Planning Domain Definition Language (PDDL), delegando lógica secuencial a planificadores simbólicos. Este enfoque garantiza coherencia lógica, validando su viabilidad en escenarios donde la flexibilidad del LLM y seguridad del PDDL son indispensables.

Palabras clave: Robótica inteligente, Planificación de tareas y movimientos, Interacción humano-robot, Robótica basada en Inteligencia Artificial, Percepción robótica.

Hybrid cognitive architecture for remanufacturing: Integrating semantic abstraction with PDDL planning

Abstract

The transition toward Industry 5.0 demands robotic systems capable of collaborating with humans, not only in structured production lines but also in value recovery processes like remanufacturing. This context presents unique challenges: high variability in incoming objects and low production cadence, which hinders traditional robotic programming. While Large Language Models (LLMs) offer semantic generalization, their application lacks the robustness required for critical disassembly sequences. This paper proposes a hybrid, modular architecture for robotic orchestration and planning in remanufacturing environments. Unlike end-to-end approaches, our proposal decouples semantic reasoning from deterministic planning. The system utilizes perception models and LLMs to abstract the unstructured environment into Planning Domain Definition Language (PDDL) files, delegating sequential logic to symbolic planners. This approach ensures task logic, validating its viability in scenarios where the flexibility of LLMs and the safety of PDDL are indispensable.

Keywords: Intelligent robotics, Task and motion planning, Human-robot interaction, AI-powered robotics, Robot perception.

1. Introducción

La industria ha evolucionado desde una visión en la que las máquinas reemplazaban a los trabajadores hacia entornos donde robots y operarios colaboran en espacios compartidos. Este cambio responde a la dificultad de los robots para operar de forma autónoma en escenarios dinámicos, impulsando el desarrollo de la Interacción Humano-Robot (HRI) para combinar la precisión robótica con la adaptabilidad humana.

Este reto es especialmente relevante en la *Remanufactura*, donde los productos llegan en condiciones desconocidas y altamente variables. A diferencia de la fabricación tradicional, el entorno no es determinista y entrenar modelos específicos para cada variante resulta inviable. Se requieren sistemas capaces de razonar sobre objetos no vistos previamente y generar planes de acción seguros.

Los Grandes Modelos de Lenguaje (LLMs) han demostrado un notable potencial para dotar a los robots de capacidades de razonamiento e interpretación de instrucciones. No obstante, su uso directo en el control robótico presenta limitaciones im-

portantes, como la falta de anclaje al entorno físico y la posible generación de acciones incompatibles con las restricciones del sistema.

Para abordar estas limitaciones, proponemos desacoplar el razonamiento semántico de la planificación determinista. Nuestra arquitectura modular utiliza el LLM como traductor semántico que, a partir de la percepción y las instrucciones del usuario, genera descripciones formales del dominio y del problema en PDDL. Estas descripciones alimentan un planificador simbólico que garantiza la coherencia y ejecutabilidad del plan. El enfoque combina flexibilidad semántica con garantías formales de planificación.

2. Trabajos Previos

2.1. Planificación mediante LLMs y Modelos VLA

El uso de LLMs para transformar lenguaje natural en secuencias de acciones ha crecido con enfoques como ProgPrompt (Singh et al., 2022), que genera programas estilo Python para descomponer objetivos de alto nivel. Otros sistemas, como RoboGPT (You et al., 2023), integran estos modelos en ROS mediante mecanismos de emparejamiento de objetos. Sin embargo, un desafío central es la falta de anclaje: los LLMs no modelan intrínsecamente las consecuencias físicas de sus actos.

Para mitigar esta limitación, SayCan (Ahn et al., 2022) combina el razonamiento del LLM con funciones de valor de bajo nivel, mientras que SayPlan (Rana et al., 2023) y SayNav (Rajvanshi et al., 2024) emplean gráficos de escena 3D para permitir búsqueda semántica y replanificación iterativa. En paralelo, los modelos de Visión-Lenguaje-Acción (VLA) como OpenVLA (Kim et al., 2024) proponen un control end-to-end directamente desde entradas visuales. Aunque efectivos en generalización semántica, estos modelos suelen operar como *cajas negras* y carecen del razonamiento a largo plazo necesario para tareas secuenciales complejas con precondiciones estrictas.

2.2. Planificación Simbólica (PDDL) y Arquitecturas Híbridas

La planificación simbólica basada en PDDL sigue siendo un pilar en la autonomía industrial por permitir sistemas correctos por construcción (Ammar and Bhiri, 2018). Para salvar la brecha entre la lógica simbólica y la ejecución física, RobMAP (Rajendran et al., 2022) integra PDDL con algoritmos de movimiento como RRT, mientras que PlanSys2 (Martín et al., 2021) y MERLIN2 (González-Santmarta et al., 2023) facilitan la gestión distribuida de planes y la ejecución reactiva en ROS2.

En entornos de manufactura, el modelado mediante ontologías permite adaptar dinámicamente los recursos robóticos disponibles (Kootbally, 2016). Frente a otros lenguajes como ASP (Jiang et al., 2019), PDDL demuestra mayor eficiencia en secuencias largas, típicas en procesos de desmontaje. Recientemente, trabajos que integran LLMs (Zhang et al., 2023) no para sustituir al planificador, sino para guiar su búsqueda mediante heurísticas, combinando capacidad generativa con validación lógica previa a la ejecución.

2.3. Brecha identificada

A pesar de estos avances, los enfoques basados en LLM y VLA ofrecen adaptabilidad pero no garantizan coherencia lógica ni cumplimiento de precondiciones físicas en secuencias críticas. Por su parte, las arquitecturas simbólicas proporcionan robustez formal pero dependen de dominios estáticos predefinidos. Existe, por tanto, un vacío metodológico en la generación dinámica de modelos verificables a partir de percepción abierta e instrucciones naturales que permita operar en entornos no estructurados con garantías de seguridad.

3. Metodología propuesta

Para dotar al sistema robótico de la adaptabilidad necesaria en procesos de remanufactura, se define un flujo de trabajo que prioriza la abstracción semántica, desde la digitalización del entorno hasta la ejecución segura mediante planificación simbólica.

3.1. Proceso de trabajo

El proceso que debe seguir una aplicación robótica que se pueda adaptar de forma dinámica a su entorno sigue el flujo que se puede observar en la Figura 1.

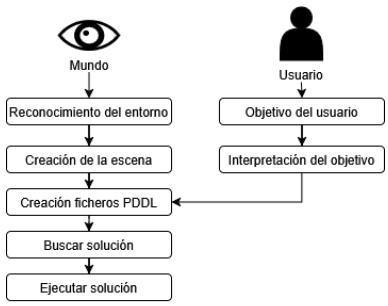

Figura 1: Pasos a seguir en una aplicación robótica adaptativa.

- *Reconocimiento del entorno*: El sistema robótico debe ser capaz de comprender dónde está y qué objetos se encuentran a su alcance. Aquí es donde entra en juego los modelos VLM a la hora de facilitar el reconocimiento de objetos sin usar modelos específicos previamente entrenados. Sin embargo, esto es efectivo si estamos buscando objetos comunes, ya que son con los que han sido entrenados. Es posible combinar la percepción generalista con modelos entrenados específicamente para la aplicación objetivo. Pensando en un ejemplo de recuperación de las celdas de una batería de un vehículo eléctrico, un VLM puede reconocer las herramientas de trabajo, pero elementos como las celdas sólo es posible percibirlas mediante modelos entrenados específicamente con esos objetos. Bajo este enfoque, solo es necesario preparar modelos para piezas específicas o extrañas. Estos modelos se ejecutan solo cuando el contexto lo requiere (por ejemplo, lanzando el modelo de celdas solo al visualizar el interior de la batería), minimizando el consumo de recursos del sistema

- *Creación de la escena*: Con la información recogida de la percepción, se reconstruye una *escena del mundo* dentro del cual se encuentra el robot, colocando los objetos dentro del entorno 3D tal y como se percibieron en el anterior paso. La escena almacena poses y relaciones espaciales que posteriormente se traducen a predicados PDDL. La escena se instancia en un simulador que permite validar geométricamente las acciones antes de su ejecución física. De esta simulación se obtendrá a su vez el estado de partida del mundo, necesario en PDDL para comenzar a buscar una solución.

- *Generación de ficheros PDDL*: El fichero de dominio (Código 1) aloja la información sobre los tipos de objetos de la escena con sus características y atributos en forma de *predicados* (qué herramienta está sujetando el robot, si el tornillo está o no atornillado a la tapa). A su vez se definen las acciones que pueden ejecutarse (se puede recoger/dejar el tornillo, o se puede desatornillar). El fichero de problema (Código 2) contiene la información sobre el *estado inicial* del mundo, indicando los *predicados* de los que partimos. Aquí es donde se define si, por ejemplo, un objeto está encima de otro, para que cuando se quiera realizar una acción sobre el objeto que esté debajo, quede definido que es necesario quitar los objetos que estén encima para poder actuar sobre él. En el fichero de problema también se encuentra la definición de la *meta*, objetivo el cual queremos conseguir. Este último vendrá dado de la petición de un usuario, explicado en el siguiente apartado.

Código 1: Ejemplo de generación de fichero de dominio

```
(define (domain remanufacturing_domain)
    (:requirements :strips :typing)
    (:types robot tool screw cover)
    (:predicates ;; Atributos de la escena
      (free ?r - robot)
      (on ?s - screw ?c - cover) ;...
    )
    (:action pickup ;; Parametros, precondiciones y
        efectos de recoger la herramienta.
    )
    (:action putdown ;; Parametros, precondiciones y
        efectos de soltar la herramienta.
    )
    (:action unscrew ;; Parametros, precondiciones y
        efectos de desatornillar.
      )
    ) )
```

Código 2: Ejemplo de generación de fichero de problema

```
(define (remanufacturing_problem)
  (:domain remanufacturing_domain)
  (:objects ;; Actor (robot) y objetos detectados
      robot1 - robot
      screw1 screw2 - screw
      screwdriver1 - tool
      cover1 - cover
  )
  (:init ;; Obtenido de la escena
      (free robot1)
      (holding-nothing robot1) ; ...
  )
  (:goal ;; No existe. Se obtiene de una peticion del
       usuario.
  ))
```

- *Interpretación del objetivo del usuario*: El sistema espera de la entrada del usuario para generar la meta. Aquí es donde entran los LLMs, permitiendo la comunicación usuario-sistema mediante Procesamiento de Lenguaje Natural (NLP). Las instrucciones provenientes del usuario deben interpretarse para generar un objetivo compatible con PDDL. Para ello se busca entre los objetos del dominio todo lo necesario para alcanzar dicho objetivo, y se consideran las acciones necesarias a tener en cuenta. Esta interpretación también hace uso de las capacidades de abstracción de los modelos de lenguaje, pudiendo interpretar las acciones de formas indirectas. El estimador de metas emplea razonamiento de sentido común mediante LLMs para inferir pasos intermedios no detectados explícitamente, como la necesidad de explorar contenedores (ej. cajas de herramientas) para localizar objetos requeridos

  Considerando un ejemplo sencillo de esta interpretación del usuario, podemos considerar la entrada del usuario *"Retira la tapa"*, solicitando al LLM que interprete la petición, adjuntando los ficheros de dominio y problema (para que acceda a los objetos y predicados). El resultado es una salida estructurada en formato JSON (Código 3), que actúa como formato intermedio controlado antes de su posterior traducción al plan ejecutable.

Código 3: Salida estructurada del LLM

```
{
  "inferred_goal": [
    "unscrewed screw1",
    "unscrewed screw2"
  ],
  "justification": "All screws attached to the cover
      must be removed before lifting it."
}
```

  La salida incluye la meta inferida y su *justificación*, que el usuario puede validar antes de la traducción a PDDL. El último paso es transformar la salida del LLM en formato PDDL y añadirlo al fichero de problema.

- *Buscar solución*: Una vez definidos todos los elementos de los ficheros PDDL se busca la solución a la meta solicitada. Previamente a la búsqueda del plan, los ficheros se validan utilizando la librería de Python *pddlpy*. Esta librería nos permite analizar el contenido de los ficheros PDDL, separando por objetos, predicados, acciones, etc. En el caso de que los ficheros generados no sean válidos, recibiremos un error, pudiendo revisarlo para buscar dónde se encuentran los problemas de generación.

- *Ejecutar solución*: El último paso consta de enviar el plan generado a un ejecutor, que se encargará a su vez de verificar que los pasos se están siguiendo de forma adecuada, o informar si ha ocurrido algún problema durante uno de ellos.

Si bien la Figura 1 ilustra el flujo lógico de resolución de tareas de forma secuencial, el sistema opera de manera concurrente. En ausencia de comandos de usuario (*sistema en reposo*), el módulo de percepción se mantiene activo en segundo plano, actualizando dinámicamente la *Escena del mundo*. Esto

garantiza que la instanciación de nuevos objetos y la modificación de sus relaciones espaciales se reflejen en tiempo real antes de iniciar cualquier planificación.

Podemos denominar *sistema en reposo* al momento en el que se está actualizando la escena con la información del mundo exterior. Tan pronto como se establece una meta, el sistema captura las condiciones iniciales (*init*) del problema. Este evento actúa como disparador del proceso de deliberación, iniciando la búsqueda de una solución simbólica, pasando a ser un *sistema en ejecución*. Ante fallos durante la ejecución, pasamos a un *sistema en fallo*, el cual notificará el problema al usuario y regresará al estado de reposo, actualizando la escena para intentar incluir los cambios que se hayan producido hasta ese momento. Todo esto puede verse de forma gráfica en la Figura 2.

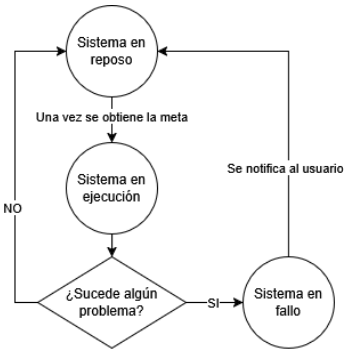

Figura 2: Diagrama del flujo para el estado del sistema adaptativo.

4. Arquitectura

4.1. Diseño modular desacoplado

Para abordar la alta variabilidad inherente a la remanufactura, hemos diseñado una arquitectura cognitiva híbrida que desacopla la percepción y la acción del razonamiento lógico. Tal como se ilustra en la Figura 3, la arquitectura se estructura en cuatro módulos:

- *Intérprete del mundo*: Procesa la información proveniente de diferentes sensores para identificar objetos y sus relaciones espaciales, actualizando continuamente la representación del entorno.
- *Intérprete de actores*: Traduce la descripción cinemática del robot a un formato compatible con PDDL. Este módulo suele ejecutarse únicamente durante la inicialización para instanciar al robot, definiendo sus acciones (e.g., tipos de agarre disponibles). Dado que la descripción cinemática del robot es estática, no requiere actualizaciones continuas, salvo que se modifiquen las herramientas o 'grippers' equipados.
- *Escena del mundo*: La información de los intérpretes converge en este módulo. No solo almacena la posición de los objetos, sino que instancia los predicados PDDL que conforman el estado inicial (init) del problema. Actúa como un entorno de simulación intermedio que permite validar geométricamente las acciones antes de su ejecución física.
- *Estimador de metas*: Este módulo actúa como puente entre el usuario y el planificador, utilizando LLMs para traducir instrucciones en lenguaje natural a objetivos PDDL válidos. Su función crítica reside en la capacidad de inferencia y enriquecimiento dinámico: cuando el dominio carece de una acción directa para cumplir la orden, el estimador utiliza conocimiento de sentido común o consulta manuales externos para descomponer la meta en subobjetivos. Por ejemplo, ante la necesidad de abrir un equipo sin una primitiva específica de *desatornillar*, el sistema puede inferir esta nueva capacidad a partir de la acción *recoger_objeto* y la presencia de un destornillador en la escena. Así, el módulo genera dinámicamente la acción necesaria, heredando las precondiciones de las tareas precedentes y asegurando que el plan resultante sea lógicamente consistente y ejecutable con las primitivas disponibles.

4.2. Infraestructura de Software

La arquitectura propuesta se plantea sobre el middleware ROS2 (Robot Operating System), aprovechando su naturaleza descentralizada para la comunicación entre módulos. Cada bloque funcional de la Figura 3 opera como un nodo independiente, garantizando la modularidad y escalabilidad del sistema. Esta descentralización habilita la ejecución concurrente de múltiples instancias de un mismo nodo, siendo esto especialmente valioso para el *intérprete del mundo*, pues permite desplegar diversos agentes que operan con modelos heterogéneos. Cada agente aporta información única a la *escena del mundo*, la cual actúa como punto de agregación para consolidar un contexto estructurado y unificado

Para la gestión de LLMs y VLMs se ha decidido usar Ollama, herramienta que facilita el alojamiento de modelos de forma local y permite realizar rápidas iteraciones entre diversos modelos sin comprometer la privacidad, algo clave en entornos industriales.

La validación de las acciones se lleva a cabo en un entorno simulado, tal como se define en el módulo de *escena del mundo*. Actualmente se está desarrollando una simulación en Webots (Webots, 2025) que sea fiel al espacio de trabajo real.

Figura 4: Entorno real (izquierda) y entorno de simulación (derecha).

4.3. Caso de Uso Propuesto Desmontaje de Tapa

Para validar la coherencia de la arquitectura propuesta, se ha definido un escenario de referencia centrado en el desmontaje de un dispositivo cerrado mediante tornillos (Figura 5). En este flujo de trabajo teórico, el ciclo comienza cuando el *intérprete del mundo* detecta visualmente los elementos de la escena (e.g., tapa, carcasa, tornillos, destornillador) y los instancia en la *escena del mundo*, traduciendo sus propiedades espaciales a predicados PDDL.

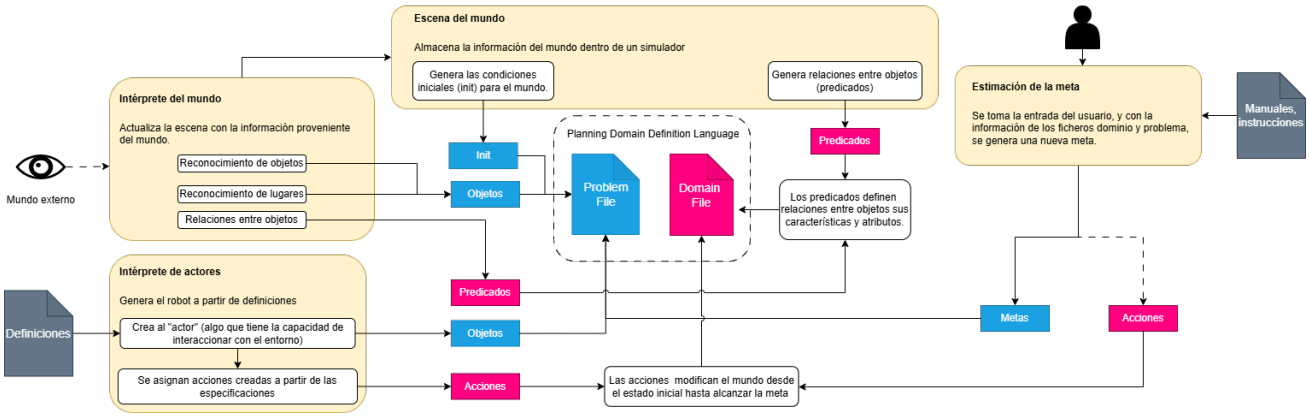

Figura 3: Esquema general de la arquitectura

Ante la instrucción del usuario "retira la tapa", el módulo de *estimación de metas* despliega su capacidad de razonamiento semántico. Dado que el dominio PDDL base podría no contener una acción directa *retirar_tapa* para un objeto atornillado, el LLM infiere, basándose en el contexto de las herramientas detectadas, que es necesario alcanzar un estado intermedio donde los tornillos hayan sido extraídos (unscrewed).

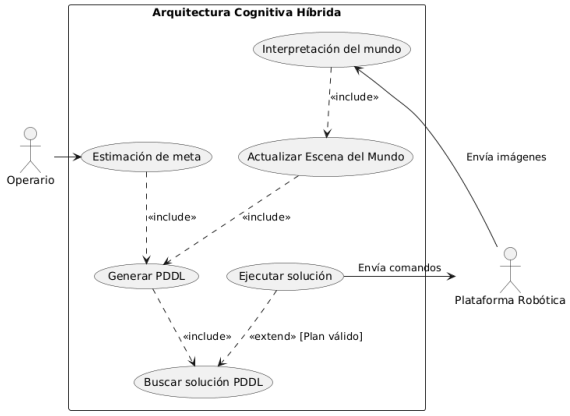

Figura 5: Diagrama del caso de uso.

Finalmente, el sistema acopla el problema PDDL resultante. El planificador simbólico es el encargado de calcular la secuencia lógica (coger destornillador → desatornillar → depositar herramienta → coger tapa). Actualmente, los esfuerzos de desarrollo se centran en lograr que esta secuencia sea validada cinemáticamente dentro del simulador antes de su paso a la ejecución física.

5. Resultados experimentales

Para analizar la viabilidad de la arquitectura híbrida propuesta, se diseñó un conjunto de experimentos orientados a evaluar tanto los módulos de percepción como el proceso de abstracción semántica y generación de planificación simbólica.

5.1. *Evaluación de VLM*

Para validar el módulo de abstracción del entorno, se llevó a cabo una evaluación comparativa de varios VLMs ejecutados en local. Cada modelo fue sometido a un conjunto común de imágenes reales del entorno de trabajo y evaluado en tres capacidades: (i) reconocimiento de objetos, (ii) abstracción del entorno (identificación del tipo de escenario) y (iii) detección de personas. Se evaluaron diversos VLMs locales mediante un dataset de imágenes reales de entornos industriales y de oficina. La métrica de rendimiento (Tabla 1) comprende la media de aciertos en tres ejecuciones por imagen sobre tres categorías: reconocimiento de objetos, abstracción del entorno y detección de operarios. En reconocimiento de objetos, se consideraron válidas respuestas semánticamente equivalentes (e.g., "allen key" y "allen wrench"). Se puede apreciar un mejor resultado para el modelo con más parámetros, aunque no es una relación lineal. Existen modelos más pequeños como MiniCPM que obtienen mejores resultados que otros con más parámetros como LLaVA (con 13b de parámetros). En el caso de detecciones de objetos extraños, deberán entrenarse modelos de forma dedicada a ese objetivo.

Tabla 1: Multimodal VLM results on object, location and human recognition

Model	Obj. (%)	Loc. (%)	Hum.(%)
LLaVA (7b)	37.39	96.00	96.00
LLaVA LLaMa3 (7b)	44.49	74.67	94.67
LLaVA (13b)	45.11	97.33	96.00
Gemma3 (4b)	16.25	50.67	58.67
MiniCPM (8b)	67.79	92.00	100.00
Mistral Small 3.1 (24b)	75.79	97.33	96.00

5.2. *Evaluación de interpretación de metas*

Para evaluar la capacidad de interpretar órdenes de usuario para generar metas de PDDL, se probaron diversos LLM frente a cuatro entornos dominio-problema, a los cuales se les había retirado el campo de meta. Estos cuatro dominios son:

- Retirar tapa atornillada (R_tapa): Presentado en el Código 1, donde tenemos una tapa cerrada por tornillos.
- Asistente de taller (Herramientas): Solicitud de herramientas al usuario, pueden estar en la mesa o dentro de una caja de herramientas.
- Retirar batería de dispositivo (R_bateria): Obtener la batería retirando cubiertas que tiene encima.
- Retirada PCB (R_PCB): Solicitud de retirada una placa electrónica, soltando previamente las conexiones.

Tabla 2: Tiempo medio de respuesta (s) en la generación de metas para diversos modelos.

Petición	Dominio	qwen2.5-coder (7b)	qwen3 (8b)	llama3.1 (8b)	llama3.2 (3b)	llama3.3 (70b)	mistral (7b)	mixtral (8x7b)
Abre la tapa	R_tapa	6.752	131.361	8.345	9.350	45.391	9.848	14.688
Ayudame con la tapa, quítala	R_tapa	8.104	43.595	9.093	11.161	69.317	11.434	17.068
Echame un cable, coge la tapa y retirala	R_tapa	8.002	128.531	9.859	10.542	70.778	9.592	31.002
Pasame los alicates	Herramientas	9.985	56.937	9.916	11.067	67.779	13.970	19.345
Traeme el martillo	Herramientas	8.237	47.9496	9.9496	8.795	66.133	13.250	20.259
Busca por ahi y dame el destornillador	Herramientas	10.411	76.413	11.954	13.231	99.424	12.267	17.301
Extract the battery	R_bateria	8.721	63.383	9.379	8.889	62.205	14.147	18.974
Open the device and remove the battery.	R_bateria	12.602	112.664	9.455	11.439	91.670	17.924	18.316
Take the battery out safely.	R_bateria	8.168	130.591	9.555	13.070	76.284	15.281	19.057
Open the enclosure and extract the PCB	R_PCB	9.831	141.976	10.078	8.807	91.743	15.678	22.530
Detach and remove the circuit board.	R_PCB	9.887	210.575	11.259	10.535	85.184	15.196	31.783
The PCB must be removed	R_PCB	7.602	140.715	9.406	12.457	80.503	12.045	18.630

Para cada dominio se definieron tres fases y se evaluó la inferencia de metas (Código 3) en 10 repeticiones por modelo, registrando tiempos y resultados. En los dos últimos dominios se utilizaron frases en inglés para analizar el efecto del idioma. En la Tabla 2 se presenta el tiempo medio de respuesta para cada uno de los diferentes modelos. No debe considerarse el tiempo de inferencia como tal, ya que es dependiente del equipo que se utilice y de los recursos disponibles, sino que debe compararse de forma relativa entre modelos.

Sin embargo, es necesario identificar si la meta generada tiene sentido y es válida. Por lo general, todos los modelos son capaces de interpretar la respuesta, pero tienen problemas a la hora de adherirse a la estructura indicada, respondiendo con el JSON correcto, a veces envuelto por otros comentarios. Estos modelos prometen buenos resultados, pero es necesario realizar una tarea más extensa de ingeniería de *prompts* para afinar los resultados. Si de lo contrario se alcanza un momento en el cual el modelo en particular genera siempre fragmentos añadidos siguiendo la misma estructura, es posible adaptar el módulo para poder recibir esta nueva respuesta y extraer las partes relevantes de la misma.

6. Conclusiones

Este trabajo presenta un sistema cognitivo híbrido para robótica de remanufactura basado en el desacoplo del razonamiento semántico (LLM) y la planificación determinista (PDDL). Esta arquitectura permite gestionar la variabilidad del entorno garantizando secuencias de acción robustas y seguras, superando las limitaciones de los modelos generativos puros.

Actualmente, se ha definido la estructura modular y el stack tecnológico (ROS 2 y Ollama). Con la integración y simulación en fase de desarrollo, el trabajo inmediato se centra en codificar las interfaces entre la estimación de metas y el planificador simbólico. El objetivo final será la validación experimental en tareas de desmontaje de objetos no vistos previamente.

Agradecimientos

Este trabajo ha sido financiado por la Unión Europea a través del programa Horizonte Europa, en virtud del acuerdo de subvención n.º 101135784 (proyecto ARISE), así como por el proyecto SOROCARE (PID2024-157671OB-I00), financiado por MICIU/AEI/10.13039/501100011033 y el Fondo Europeo de Desarrollo Regional (FEDER, UE).

Referencias

Ahn, M., Brohan, A., Brown, N., et al., 2022. Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691.
DOI: 10.48550/arXiv.2204.01691

Ammar, S., Bhiri, M. T., 2018. Automatic planning: From event-b to pddl. In: Proceedings of the International Conference on Automatic Planning. pp. 247–254.
DOI: 10.1007/978-3-030-02852-7$_2$1

González-Santmarta, M., Rodríguez-Lera, F. J., Fernández-Llamas, C., et al., 2023. Merlin2: Machined ros 2 planing. Software Impacts 15, 100477.
DOI: 10.1016/j.simpa.2023.100477

Jiang, Y., Zhang, S., Khandelwal, P., et al., 2019. Task planning in robotics: An empirical comparison of pddl- and asp-based systems. Frontiers of Information Technology & Electronic Engineering 20 (3), 363–373.
DOI: 10.1631/FITEE.1800514

Kim, M. J., Pertsch, K., Karamcheti, S., et al., 2024. Openvla: An open-source vision-language-action model. arXiv preprint arXiv:2406.09246.
DOI: 10.48550/arXiv.2406.09246

Kootbally, Z., 2016. Industrial robot capability models for agile manufacturing. Industrial Robot: The International Journal of Robotics Research and Application 43 (5), 481–494.
DOI: 10.1108/IR-02-2016-0071

Martín, F., Clavero, J. G. G., Matellán, V., et al., 2021. Plansys2: A planning system framework for ros 2. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 9742–9749.
DOI: 10.1109/IROS51168.2021.9636544

Rajendran, G., V., U., O'Brien, B., 2022. Unified robot task and motion planning with extended planner using ros simulator. Journal of King Saud University – Computer and Information Sciences 34 (9), 7468–7481.
DOI: 10.1016/j.jksuci.2021.07.002

Rajvanshi, A., Sikka, K., Lin, X., et al., 2024. Saynav: Grounding large language models for dynamic planning to navigation in new environments. In: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS). pp. 464–474.
DOI: 10.1609/icaps.v34i1.31506

Rana, K., Haviland, J., Garg, S., et al., 2023. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. arXiv preprint arXiv:2307.06135.
DOI: 10.48550/arXiv.2307.06135

Singh, I., Blukis, V., Mousavian, A., et al., 2022. Progprompt: Generating situated robot task plans using large language models. arXiv preprint arXiv:2209.11302.
DOI: 10.48550/arXiv.2209.11302

Webots, 2025. http://www.cyberbotics.com. Open-source Mobile Robot Simulation Software.
URL: http://www.cyberbotics.com

You, H., Ye, Y., Zhou, T., et al., 2023. Robot-enabled construction assembly with automated sequence planning based on chatgpt: Robogpt. Buildings 13 (7), 1772.
DOI: 10.3390/buildings13071772

Zhang, Y., Wang, Z., Zhang, S., et al., 2023. Boosting robot intelligence in practice: Enhancing robot task planning with large language models. In: Proceedings of the International Conference on Robotics and Automation Engineering (ICRAE). pp. 90–94.
DOI: 10.1109/ICRAE59816.2023.10458574