Flexible Remanufacturing using AI and Advanced Robotics for Circular Value Chains in EU Industry

Grant agreement nº: 101138415

Call identifier: HORIZON-CL4-2023-TWIN-TRANSITION-01-04
Factory-level and value chain approaches for remanufacturing
(Made in Europe Partnership) (IA)

D5.1: Robot Skills & Flexible Production Modules for Remanufacturing – Initial Prototype

Deliverable nature:	Demonstration
Dissemination level:	Sensitive
Delivery date:	27/06/2025
Version:	V1.0
Lead Author:	DTI
Contributors:	LMS, TECNALIA, IIT, COMAU
Reviewer:	LMS, INESC

Funded by the European Union

This project has received funding from the European Union's Horizon Europe Framework Programme under grant agreement nº 101138415.

Executive Summary

D5.1: Robot Skills & Flexible Production Modules for Remanufacturing – Initial Prototype, presents the first iteration of a comprehensive framework designed to enable flexible and adaptive remanufacturing operations through advanced robotics and AI technologies. The document encompasses four interconnected technological pillars that form the foundation of the RENÉE approach to automated remanufacturing.

The Robot Skills Library (RSL) provides a modular, extensible collection of robot capabilities that can be easily deployed, reconfigured, and adapted to handle diverse remanufacturing tasks. By encapsulating robotic functionalities as discrete, reusable "skills," the RSL significantly reduces programming complexity and enables rapid adaptation to different product variants and core conditions.

The Flexible Production Modules section details the architecture and implementation of reconfigurable manufacturing resources designed for remanufacturing environments. This includes the Resource Control Framework, that orchestrates task execution hardware integration interfaces that ensure interoperability across diverse equipment and robotic resources. The requirements for the robot selection for each use case are also presented in this section.

The AI-Enhanced Perception System addresses the critical challenge of core variability in remanufacturing by implementing advanced vision and sensing capabilities. The perception framework enables robots to identify, locate, and assess components in varying conditions, supporting critical functions such as multiple parts detection, action recognition, visual condition assessment, and precise manipulation.

The Intelligent End Effectors and Smart Mechatronics section presents innovative gripper designs and actuation systems capable of manipulating the diverse components encountered in remanufacturing scenarios. These solutions incorporate advanced sensing and control capabilities to handle both rigid and flexible materials with precision and adaptability.

This deliverable contributes directly to WP5's objectives by establishing the technological foundation for human-centric production systems that leverage advanced robotics and AI to accommodate resilient remanufacturing despite uncertainties in supply and demand. The initial prototypes presented here will be further refined and integrated in subsequent project phases to deliver comprehensive remanufacturing solutions across the four industrial use cases.

Disclaimer

Table of Contents

Abbreviations List

Abbreviation	Definition
AAS	Asset Administration Shell
ADS	Automation Device Specification
AI	Artificial Intelligence
BT	Behaviour Tree
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
COIGAN	Controllable Object Inpainting with Generative Adversarial Networks
D&C	Dissemination & Communication
DoF	Degree of Freedom
DPP	Digital Product Passport
DT	Digital Twin
EU	European Union
FOV	Field-of-View
GPT	Generative Pre-trained Transformer
HAL	Hardware Abstraction Layer
HTS	Hierarchical Token-Semantic
HW	Hardware
ICRA	International Conference on Robotics and Automation
IEEE	Institute of Electrical and Electronics Engineers
IK	Inverse Kinematics
IO	Input/Output
ISO	International Organization for Standardization
JSON	JavaScript Object Notation

LLM	Large Language Model
OGN	OmniGraph Node
OSI	Operator Support Interfaces
PCB	Printed circuit boards
PLC	Programmable Logic Controller
PSD	Product State Diagnosis
PSO	Process Scheduler and Orchestrator
Q-Former	Querying Transformer
ROI	Region of Interest
ROS	Robot Operating System
RSL	Robot Skills Library
TCP	Tool Centre Point
TF	Transformation
URDF	Unified Robot Description Format
VLM	Vison Language Model
WP	Work Package
XML	Extensible Markup Language
YAML	Yet Another Markup Language
YOLO	You Only Look Once

List of Figures

List of Tables

1 Introduction

1.1 Objective of the Deliverable

The objective of deliverable D5.1, "Robot Skills Flexible Production Modules for Remanufacturing – Initial Prototype," is to present the initial development and integration of modular, adaptable robotic skills and flexible production modules designed to address the challenges of remanufacturing within circular value chains. This deliverable aims to demonstrate how advanced robotics, AI-driven perception, and intelligent end-effectors can be leveraged to automate and optimise key remanufacturing processes—such as disassembly, inspection, and component handling—across diverse industrial scenarios.

Specifically, D5.1 documents the creation of a reusable robot skills library, the implementation of flexible control frameworks, and the integration of AI-enhanced perception systems, all validated in the RENÉE project's four pilot use cases (household appliances, electric motors, robotics, and bicycles). The deliverable supports the overarching goal of WP5: enabling resource-level autonomy and adaptability in remanufacturing environments, thus facilitating the transition from manual, labour-intensive processes to scalable, efficient, and human-centric automated solutions. The outcome of this deliverable lay the groundwork for subsequent refinement and large-scale deployment of these technologies in the final project phase.

1.2 WP5 Architecture

The diagram below illustrates the architecture for WP5 (Figure 1) and its integration within the overall RENÉE project framework.

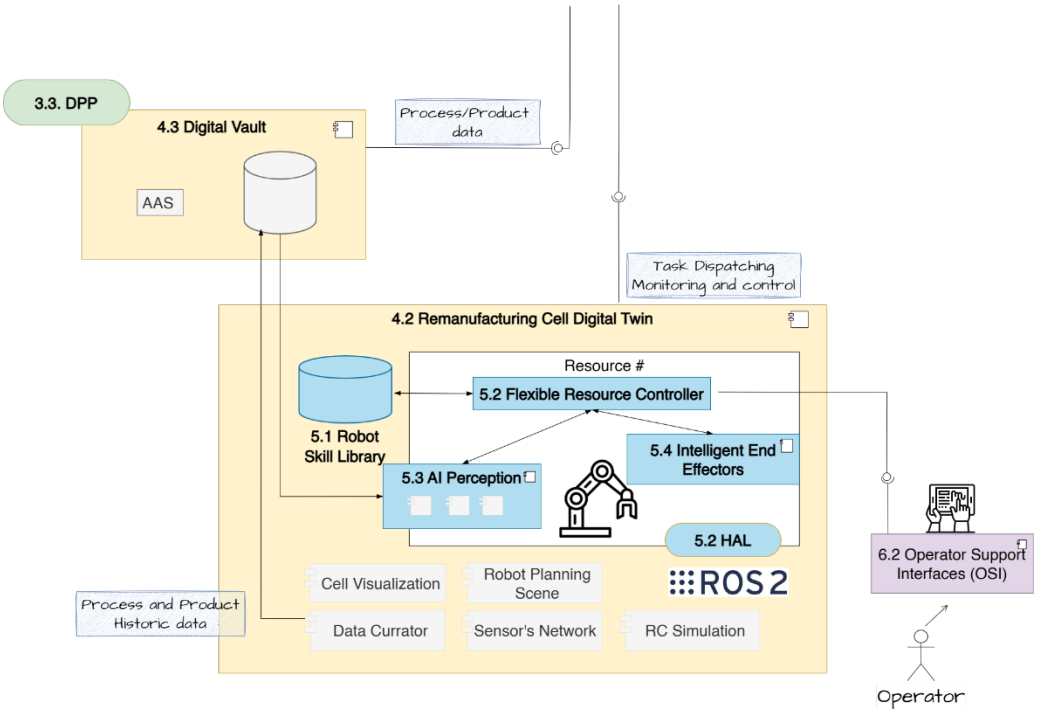

Figure 1: RENÉE Remanufacturing Cell Architecture

The architecture is centred around the robotic resources and their control systems, which are developed within WP5 and highlighted in blue. These modules operate on the ROS2 framework and include specific results from the WP5 tasks:

- **T5.1 Robot Skill Library**: A repository of reusable, modular skills (e.g., unscrewing, gripping) that can be configured for various remanufacturing tasks.
- **T5.2 Flexible Resource Controller**: The central control unit that interprets task requests, selects appropriate skills from the library, and uses perception data to command the robot and its end-effectors.
- **T5.3 AI Perception**: AI-driven modules for object detection, localization, and condition assessment, which adapt to core variability by processing "Product state" data from the WP4 Digital Vault.
- **T5.4 Intelligent End Effectors**: Adaptable grippers and tools designed for complex manipulation tasks.

These WP5 modules are enclosed within the T4.2 Remanufacturing Cell Digital Twin (shown in yellow), a component of WP4. The Digital Twin provides a virtual representation of the robotic cell, offering services such as cell visualization, robot planning, and simulation. It also receives process and product data from the T4.3 Digital Vault, which is informed by the Digital Product Passport (DPP) from WP3.

For effective human-robot collaboration, the architecture includes a crucial interface between WP5 and WP6. The T5.3 Flexible Resource Controller connects directly to the T6.1 Operator Support Interfaces (OSI), allowing human operators to monitor, control, and interact with the remanufacturing workstation seamlessly. This ensures that the automated system remains human-centric and supports the upskilling of the workforce.

2 Overview of Robot Skills Library (RSL)

2.1 Overview of Robot Skills Library

The Robot Skills Library (RSL) serves as a cornerstone technology within the RENÉE framework, designed to address the inherent complexity and variability in remanufacturing operations. The concept behind RSL is to create a modular, extensible collection of robot capabilities that can be easily deployed, reconfigured, and adapted to handle the diverse tasks encountered in remanufacturing scenarios across multiple industrial sectors.

At its core, the RSL provides a standardized approach to defining and implementing robotic capabilities as discrete, reusable “skills” that encapsulate specific functionalities required for remanufacturing tasks such as disassembly, inspection, cleaning, and material handling. Each skill represents a self-contained unit of robot behaviour that can be parameterized, sequenced, and combined with other skills to create complex operations without requiring extensive reprogramming or expert knowledge.

2.1.1 Concept and Design

The RSL architecture is built upon three fundamental principles:

- **Modularity:** Skills are designed as independent, interchangeable modules that can be combined in various configurations to accomplish different remanufacturing tasks. This modularity enables rapid adaptation to different product variants and core conditions.
- **Abstraction:** The library abstracts the complexity of robot programming by providing high-level interfaces that shield users from low-level implementation details. This makes robot programming more accessible to operators without specialized robotics expertise.
- **Extensibility:** The RSL is designed to be easily extended with new skills as remanufacturing requirements evolve, allowing for continuous improvement and adaptation to new challenges.

The skills within the library are implemented as OmniGraph nodes in NVIDIA Isaac Sim software, leveraging its powerful simulation and robotics capabilities. This implementation approach enables skills to be thoroughly tested in a virtual environment before deployment on physical robots, reducing development time and potential risks. The skills can be parametrized through intuitive interfaces, allowing for quick adaptation to specific task requirements and product variations.

By providing this structured approach to robot programming, the RSL significantly reduces the engineering effort required to implement remanufacturing processes, making automation more accessible and economically viable even for small-batch and high-variability scenarios typical in remanufacturing operations.

2.2 List of Skills per Use Case (TECNALIA, TF-CC, IIT, STAM)

The implementation of robot skills within the RENÉE framework follows a systematic approach that maps remanufacturing processes to specific robotic capabilities. To illustrate the comprehensive scope of remanufacturing operations that the Robot Skills Library addresses, Figure 2 presents the complete remanufacturing process line, showing the sequential stages from initial product assessment through final quality control and output.

This process flow demonstrates the complexity and variability inherent in remanufacturing operations, highlighting the need for flexible, adaptable robot skills that can handle diverse tasks across multiple process stages. The following subsections detail how specific skills are defined and implemented for each of the four RENÉE use cases.

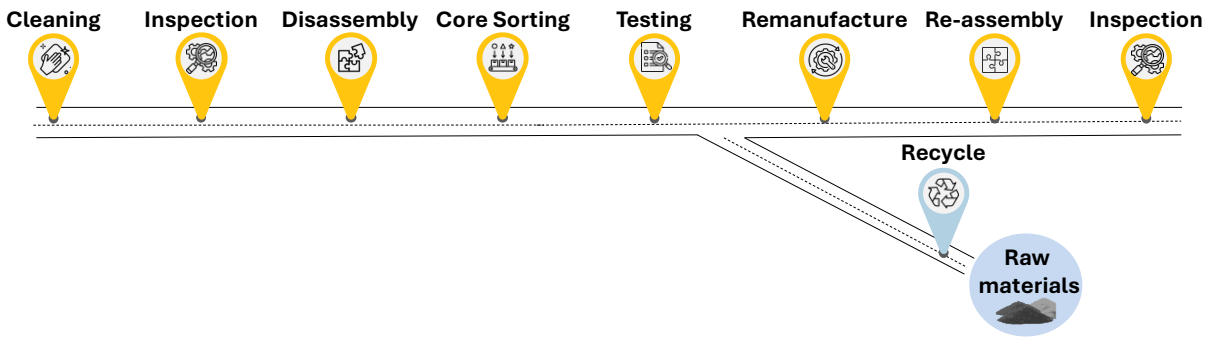

Figure 2 Remanufacturing Process line sequence

2.2.1 Skill definition in RENÉE

In the RENÉE framework, a robot skill is defined through a structured taxonomy with the following key attributes:

1. **Process Context**: Each skill belongs to a specific remanufacturing process category (e.g., Disassembly, Inspection, Cleaning, Core Sorting)
2. **Process Title**: Identifies the specific process within the broader category (e.g., "Enclosure plate removal" within Disassembly)
3. **Task ID**: A unique identifier for the specific task (e.g., "1.1", "2.3")
4. **Task Title**: A descriptive name of the specific operation (e.g., "Engine grasping by industrial robot arm")
5. **Action Type / Skill**: The fundamental robot capability required (e.g., "Grasp", "Unscrewing", "Release", "Inspection", "Move")
6. **Resource**: The specific hardware resource that executes the skill (e.g., "FANUC robot", "UR robot", "Operator")

This structured definition provides a standardized way to represent robot capabilities across different use cases, enabling:

1. Clear mapping between remanufacturing processes and required robot capabilities
2. Modular composition of complex operations from fundamental skills
3. Appropriate resource allocation based on skill requirements
4. Consistent implementation across different industrial applications (Electric motors, Household Appliances, Robots, Bicycles)

The following tables demonstrate how this skill definition approach is applied consistently across the different industrial use cases, with each table showing the specific skills required for that particular remanufacturing scenario while maintaining the same definitional structure.

Process	Process Title	Task ID	Task Title	Action Type / Skill
Disassembly	Enclosure plate removal	1.1	Engine grasping by industrial robot arm	Grasp
		1.2	Unscrewing bolting joint	Unscrewing
		1.3	Enclosure plate removal	Release
Disassembly	Temperature sensor and stator phases connectors removal	2.1	Sensor and connector unplugged	Release
Disassembly	Inverter removal	3.1	Engine positioning	Move
		3.2	Inverter cover unscrewing	Unscrewing
		3.3	Inverter removal	Release
		3.4	Inverter removal inspection	Inspection
Inspection	Inverter inspection	4.1	Electrical connections inspection	Inspection
		4.2	Inverter components status	Inspection
Disassembly	Carbon brush removal	5.1	Engine positioning	Move
		5.2	Carbon brush removal	Release
Disassembly	Rotor removal	6.1	Insert engine in press	Release
		6.2	Engine rotor removal	Remove
Disassembly	Stator removal	7.1	Engine remove from press	Grasp
		7.2	Place engine in oven	Release
		7.3	Remove stator from housing	Remove
Disassembly	Magnets removal	8.1	Remove the stacks from the rotor shaft	Remove
		8.2	Place stacks on tooling	Move
		8.3	Remove first cover plates from stack edges (upper/down sides)	Remove
		8.4	Remove the magnets from stacks	Remove

Table 1: Electric Motors Remanufacturing Skills.

Process	Process Title	Task ID	Task Title	Action Type / Skill
Inspection	Inspection & Identification	1.1	Visual Inspection	Inspection
		1.2	Refrigerator scanning	Scanning
Disassembly/ Core sorting	Upper Door Recovery	2.1	Cover Unclamping	Grasp
		2.2	Cover Storage	Release
		2.3	Electrical Connector Unplugging/Cutting	Unplug/Cut
		2.4	Unscrewing	Unscrewing
		2.5	Hinge Removal	Grasp
		2.6	Hinge Storage	Release
		2.7	Door's Grasping	Grasp
		2.8	Door's Storage	Release
Disassembly/ Core sorting	PCB Recovery	3.1	Cover Unscrewing	Unscrewing
		3.2	Cover Removal	Grasp
		3.3	Cover Storage	Release
		3.4	Connectors' Cutting	Cut
		3.5	PCB Removal	Grasp
		3.6	PCB's Storage	Release
Disassembly/ Core sorting	Fan Recovery	4.1	Interior Shelves Removal	Grasp
		4.2	Interior Shelves Storage	Release
		4.3	Fan Cover Removal	Grasp
		4.4	Fan's Cover Storage	Release

Table 2: ARCELIK - Household Appliances Remanufacturing Skills

Process	Process Title	Task ID	Task Title	Action Type / Skill
Navigation	Initial Mapping	1.1	Initial mapping	Mapping
Navigation	Navigate	2.1	Navigate to ROI	Navigate
Cleaning	Initial Cleaning	3.1	Select and change the suitable tool	Tool Changing
		3.2	Grasp the cleaner	Grasp
		3.3	Locate the arm at the dirt position	Move
		3.4	Activating the cleaner	Tool Activation
		3.5	Release the cleaner	Release
Inspection	Inspection	4.1	Navigate to ROI	Navigate
		4.2	Move the camera along the selected parts	Move
Disassembly/ Core sorting	Disassembly of Group X X = [transversal axis, Extraction axis, vertical axis]	5.1	Select and change the suitable tool	Tool Changing
		5.2	Axis fixing plate unscrewing	Unscrewing
		5.3	Axis fixing plate removal	Grasp
		5.4	Axis fixing plate removal	Release
		5.5	Motor fixing plate unscrewing	Unscrewing
		5.6	Change to normal end-effector	Tool Changing
		5.7	Grasp the motor fixing plate	Grasp
		5.8	Move to the storage	Move
		5.9	Release the motor fixing plate	Release
		5.10	Change the tool to screwdriver	Tool Changing
		5.12	Sliding rail unscrewing	Unscrewing
		5.13	Change to normal end-effector	Tool Changing
		5.14	Sliding rail removal	Grasp
		5.15	Sliding rail removal	Release
		5.16	Change the tool to screwdriver	Tool Changing
		5.17	Belt fixing plate unscrewing	Unscrewing
		5.18	Change to normal end-effector	Tool Changing
		5.19	Grasp the belt fixing plate	Grasp
		5.20	Move to the storage	Move
		5.21	Release the belt fixing plate	Release
Disassembly/ Core sorting	Disassembly of Electrical Cabinet	6.1	Change the tool to screwdriver	Tool Changing
		6.2	Unscrew the cabinet parts	Unscrewing
Inspection	Final Inspection of Dismounted Parts	7.1	Change to normal end-effector	Tool Changing
		7.2	Move the camera along the selected parts	Move

Table 3 Robotics Remanufacturing Skills

2.2.2 Bicycles

Process	Process Title	Task ID	Task Title	Action Type / Skill
Visual inspection	Visual inspection of different bike components	1.1	Cleaning	Inspection
		1.2	Completeness	Inspection
		1.3	Visual condition	Inspection
		1.4	Wheels	Inspection
		1.5	Brakes	Inspection
		1.6	Transmission	Inspection
		1.7	Crankset and pedals	Inspection
Mechanical Inspections	Frame and forks inspection	2.1	Visual inspection of frame and forks	Inspection
		2.2	Bike frame segmentation	Scanning
		2.3	Thermographic inspection of frame and forks	Trajectory computation and path following
				Carbon defects analysis
Mechanical Inspections	Suspension forks and shock absorber inspection	3.1	Position the bike on the load cell and apply the necessary sensors	
		3.2	Forks compression	
		3.3	Generation and analysis of the response curve of the forks	Data analysis

Table 4 DECATHLON - Bicycles Remanufacturing Skills

3 Development of Robot Skills

3.1 Skill Guide – How to set up your first skill

This guide provides step-by-step instructions for setting up and developing robot skills within the RENÉE framework. Designed for both novice and experienced developers, it walks through the process of installing the necessary components, creating new skills, and integrating them into remanufacturing workflows. By following this guide, users can quickly begin building custom robot capabilities that address the unique challenges of remanufacturing operations. The modular approach presented here enables rapid development and deployment of robot skills while minimizing programming complexity and development time. Whether you're implementing basic movement skills or complex multi-step operations, this guide will help you establish the foundation for effective robot programming in remanufacturing applications.

3.1.1 Installation

1. **Clone or Copy the Extension**
   - Place the com.dti.nodes folder inside your IsaacSim Extension directory (e.g., IsaacSim/Extension/com.dti.nodes).

2. **Register the Extension in IsaacSim**
   - Open IsaacSim.

- Go to Window > Extensions.
- Click the gear icon and select Add Extension Path..
- Add the path to your com.dti.nodes directory.
- Enable the extension by toggling it on in the Extensions window.

3. **Dependencies**
   - Ensure all Python dependencies required by your nodes are installed. Most dependencies are included with IsaacSim, but custom requirements should be installed via pip.

3.1.2 Adding New Nodes (“Skills”)

To add a new node (skill) to this extension, follow the steps below. This guide walks you from a basic template to a finished node.

3.1.2.1 Node Development Guide: From Template to Node

1. Copy the Basic Node Template
   - Create a new Python file in:

     `com/dti/nodes/ogn/python/nodes/`

   - Use the following template as a starting point:

```
# Basic OGN Node Template
import omni.graph.core as og

class MyNodeInternalState:
    def __init__(self):
        # Initialize any state variables here
        pass

    def custom_reset(self):
        # Reset state if needed
        pass

class MyNode:
    @staticmethod
    def internal_state():
        return MyNodeInternalState()

    @staticmethod
    def compute(db) -> bool:
        # Access inputs: db.inputs.<inputName>
        # Set outputs: db.outputs.<outputName>
        # Example: db.outputs.result = db.inputs.valueA +
db.inputs.valueB
        return True

    @staticmethod
    def release_instance(node, graph_instance_id):
        # Cleanup if needed
        pass
```

Figure 3: Basic OGN Node Template

- Rename MyNode and MyNodeInternalState to match your node's purpose.

2. Create the OGN(OmniGraph Node) Definition

- Add a corresponding .ogn file in the same directory. This file defines the node's interface for the OmniGraph system.
- Use an existing .ogn file as a template (e.g., OgnIsaacCartArticulationController.ogn).
- Update the class name and input/output definitions as needed.

3. Implement Node Logic

- Open your new Python node file and locate the `compute` method. This is where you implement your node's main logic.
- Use the `db` (database) object to access node inputs and set outputs:
  - Access inputs: `db.inputs.<inputName>`
  - Set outputs: `db.outputs.<outputName>`
- Example usage:

```
db.outputs.result = db.inputs.valueA + db.inputs.valueB
```

- Use the `internal_state` class to store persistent state between graph evaluations. This is useful for keeping track of variables or resources that should persist across multiple calls to compute.
- Add any necessary imports or helper functions at the top of your file to support your node's logic.
- If your node needs to perform cleanup (e.g., releasing resources), implement the `release_instance` method.
- For debugging, you can use `print()` statements or the `carb.log_info()/carb.log_warn()` functions if available in your environment.
- If your node requires initialization or reset logic, use the `__init__` and `custom_reset` methods in your internal state class.
- Make sure to return `True` from `compute` if the node executed successfully, or False if there was an error.
- Test your node in IsaacSim by connecting it in an OmniGraph and verifying its behavior with different input values.

Node Lifecycle and Function Call Order:

- `__init__` (in your internal state class): Called when the node is first created or the graph is loaded. Use this to initialize persistent state variables.
- `custom_reset` (in your internal state class): Called when the graph or node is reset. Use this to reset any state variables to their initial values.
- `internal_state` (static method): Called by the OmniGraph system to create or retrieve the internal state object for the node instance.
- `compute` (static method): Called every time the node is evaluated in the graph (e.g., when inputs change or the graph is executed). This is where you implement the main logic of your node.
- `release_instance` (static method): Called when the node instance is being destroyed or the graph is unloaded. Use this to clean up resources or perform any necessary teardown.

4. (Optional) Add to Category

- If you want your node to appear in a specific category in the node browser, update:

`com/dti/nodes/ogn/python/nodes/config/CategoryDefinition.json`

- Add your node's class name to the appropriate category.

5. (Optional) Update Documentation

- Add a description of your new node to the docs/README.md or docs/CHANGELOG.md as needed.

6. Reload the Extension

- In IsaacSim, disable and re-enable the extension, or restart IsaacSim, to load your new node.

3.2 Example skill – Move

3.2.1 Overview - IsaacCartArticulationController Node

The `IsaacCartArticulationController` is a custom OGN ("skill") for NVIDIA *Isaac Sim* that enables advanced Cartesian control of articulated robots. It allows you to command a robot's end-effector (TCP) to move to a desired position and orientation in 3D space, handling the necessary inverse kinematics and joint interpolation automatically. This node is designed for use with robots described by URDF and Lula YAML files and supports smooth trajectory generation for precise manipulation tasks.

3.2.2 What It Does

- Receives Cartesian position and orientation commands for a robot's end-effector (TCP).
- Computes the required joint positions using inverse kinematics (IK) with Lula and IsaacSim's kinematics solvers.
- Interpolates the motion to create smooth, multi-step trajectories.
- Applies the computed joint actions to the robot in the simulation.
- Tracks whether the target has been reached and exposes this as an output.

3.2.3 How to Use in IsaacSim

1. Add the Node to Your OmniGraph

- Open IsaacSim and load your stage with the robot you want to control.
- Open the OmniGraph window.
- Add the `IsaacCartArticulationController` node to your graph (it may appear under a custom or extension category).

2. **Configure Node Inputs**
    - Set the following required inputs:
        - `robotPath`: The prim path to your robot in the stage (e.g., /World/Robot).
        - `urdfPath`: The file path to your robot's URDF file.
        - `descriptionPath`: The file path to your Lula YAML robot description.
        - `tcpPath`: The prim path to the robot's end-effector (TCP) in the stage.
        - `positionCommand`: A 6-element array specifying the target [x, y, z, roll, pitch, yaw] for the TCP in world coordinates.
    - (Optional) Set `jointNames` if you want to control specific joints.
3. **Connect Outputs**
    - The node provides an output `targetReached` (bool) that indicates when the robot has reached the commanded pose.
4. **Typical Workflow**
    - Send a new `positionCommand` to move the robot's TCP.
    - Monitor `targetReached` to know when the motion is complete.
    - Update the command as needed for new targets.
5. **Tips**
    - Ensure your robot is properly imported and configured in the stage.
    - The node requires valid URDF and Lula YAML files for your robot.
    - For best results, use with robots that have a well-defined TCP and kinematic chain.

3.2.4 Example

Suppose you have a Fanuc robot in your stage at /World/Fanuc and want to move its TCP to a new pose:

- Set `robotPath` to `/World/Fanuc`
- Set `urdfPath` to the path of your robot's URDF file (e.g., `.../crx25ia.urdf`)
- Set `descriptionPath` to the Lula YAML file (e.g., `.../Fanuc_CRX25ia_Lula_Description.yaml`)
- Set `tcpPath` to the prim path of the TCP (e.g., `/World/Fanuc/tool0`)

- Set `positionCommand` to `[x, y, z, roll, pitch, yaw]` (desired pose)

The node will compute the necessary joint actions and move the robot accordingly.

3.3 Example skill – Core Sequencer

3.3.1 Overview – MultiplexerDouble node

The `MultiplexerDouble` is a custom OGN ("skill") for NVIDIA IsaacSim that allows you to dynamically select between multiple input arrays of doubles (float64) and route the selected input to the output. This node is useful for scenarios where you want to switch between different data sources or control signals in your simulation graph, based on external conditions or logic.

3.3.2 What It Does

- Dynamically creates a configurable number of double array inputs (`input0, input1, ..., inputN`).
- For each input, provides a corresponding selector input (`selector0, selector1, ..., selectorN`) that can be triggered to select that input.
- When a selector is activated, the corresponding input array is routed to the output.
- Outputs the selected array as both an array (`array_output`) and as a string (`string_output`).

3.3.3 How to Use in IsaacSim

1. **Add the Node to Your OmniGraph**
    - Open IsaacSim and your desired stage.
    - Open the OmniGraph window.
    - Add the `MultiplexerDouble` node to your graph (it may appear under a custom or extension category).
2. **Configure Number of Inputs**
    - Set the "`num_of_inputs`" input attribute to the number of input arrays you want to multiplex.
    - The node will automatically create the corresponding "`inputN`" and "`selectorN`" attributes.
3. **Connect Inputs and Selectors**
    - Connect your data sources to the `input0, input1, ..., inputN` attributes.
    - Use the `selector0, selector1, ..., selectorN` execution attributes to control which input is currently selected. Only one selector should be enabled at a time.
4. **Connect Outputs**
    - The node provides two outputs:

- `array_output`: The currently selected input array (as a double array).
- `string_output`: The currently selected input array, converted to a string (for debugging or display purposes).

5. **Typical Workflow**
    - Set `num_of_inputs` to the desired number of sources.
    - Connect your input arrays and control the selectors to switch between them.
    - Use the outputs in downstream nodes or for visualization.

- **Example**

Suppose you want to switch between two different control signals for a robot:

- Set `num_of_inputs` to 2.
- Connect your first control signal to `input0` and the second to `input1`.
- Use logic in your graph to enable `selector0` or `selector1` as needed.
- The output will always reflect the currently selected input.

3.4 Example OmniGraph node

In Figure 4 showing the integration of core robot skills for remanufacturing. The graph demonstrates how multiple “Make Array” nodes generate command arrays that are routed through a Multiplexer (double) node for dynamic selection. The selected commands are sent to the Cartesian Articulation Controller to control robotic motion, while a Multigate node sequences additional actions such as surface gripper operation and playback control. This architecture enables flexible task execution and modular orchestration of robot behaviours within the simulation environment.

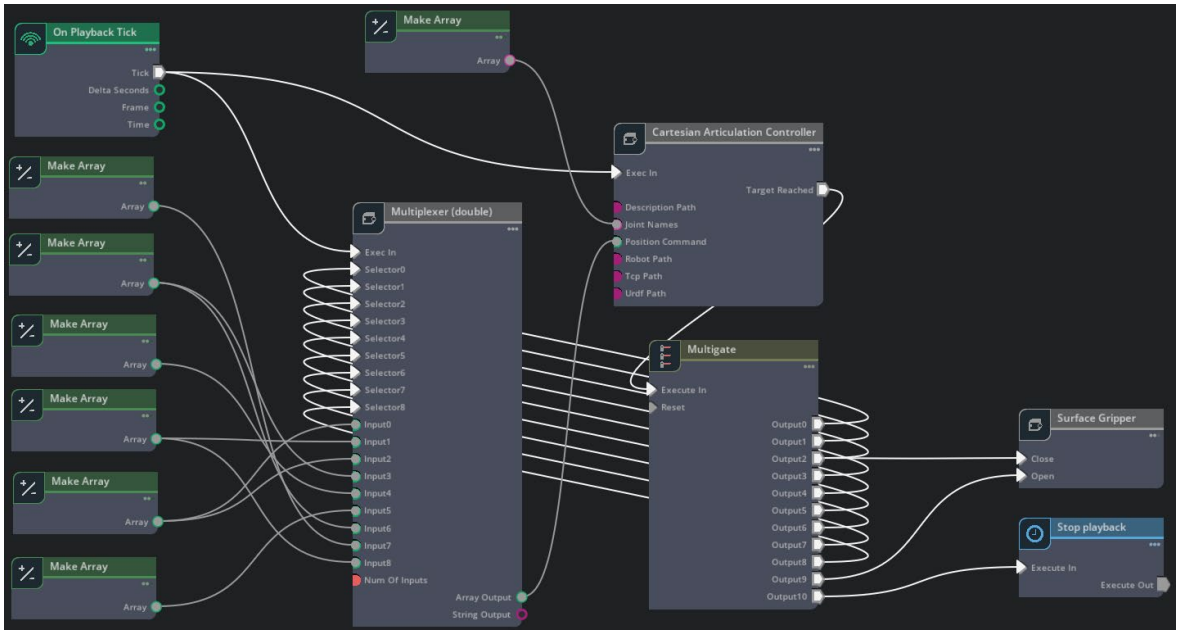

Figure 4: Example OmniGraph node.

3.5 Robot Skill Reprogramming

Remanufacturing requires the disassembly and removal of multiple parts and elements during the process. One feature of future remanufacturing lines is the retrieval of products manufactured over the last few years or decades for the retrieval of core elements. It is foreseen that during this time period, these products will evolve in design and assembly procedures, resulting in a product family with similar characteristics but differences in shape and disassembly steps (e.g. different bolt positions, different connector types…). Therefore, it is mandatory to anticipate these future issues, creating mechanisms to reprogram the robot skills for a quick adaptation of the robot programs to the diverse products received for remanufacturing.

3.5.1 Onsite Reprogramming Tools

The posed remanufacturing use cases raise the need to adapt the robot skills to a product family with a similar disassembly procedure but with slight differences in product shape and component type or placement. In some of the use cases, it will be feasible to use product CAD information to adapt the robot skills to the received product variant, although this information might be unavailable in some scenarios. Therefore, the capability to reprogram and teach the specific features of the received product variant in the disassembly robotic cell in a simple and intuitive way is essential for future remanufacturing stations.

Onsite robot reprogramming is a suitable alternative, offering a simple and intuitive way to adapt the robot skills to the specific features of the product variant (e.g. teaching the screw positions, connector removal manoeuvre…). As the majority of the use cases tackled in RENÉE project require unscrewing multiple bolts, and their placement might change in different product variants, the unscrewing task has been selected as an example of a re-programmable task that should be addressed to facilitate the program adaptation.

In this sense, the onsite reprogramming requires two main functionalities:

- An intuitive way to manage the robot and move the tool to the task position (unscrewing position in this case). In order to build a flexible tool, suitable to the different robot setups and layouts, the proposed solution relies on the use of a wireless 6D joystick and a tablet to move the robots and guide the reprogramming procedure. It allows simple teleoperation using portable devices that can be easily deployed in the different setups posed in the project.

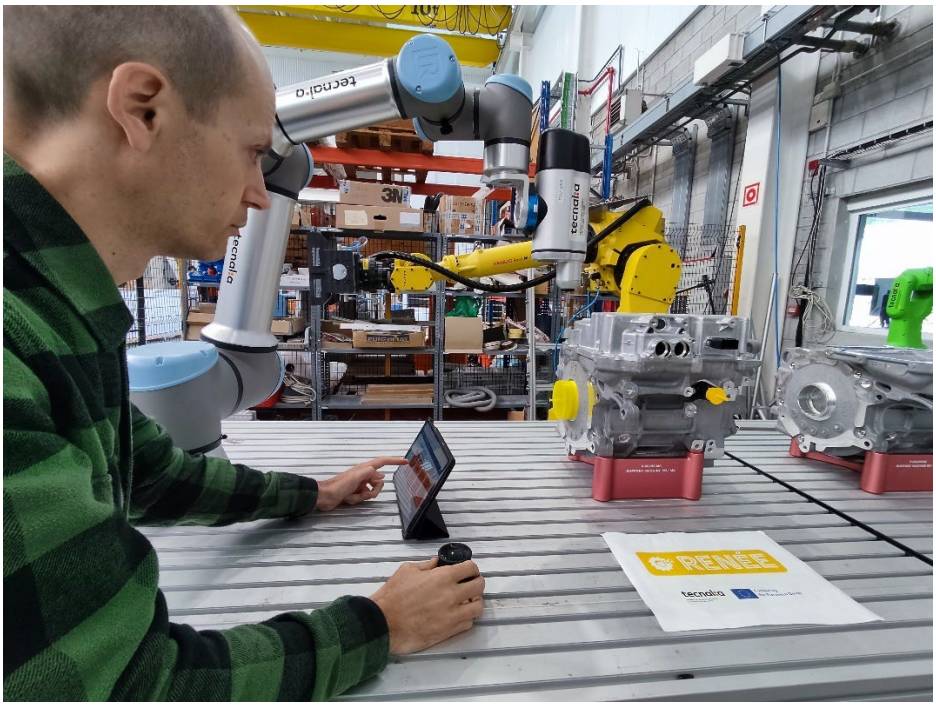

Figure 5: Programming of unscrewing positions

- The second part is the storage of the robot pose information in a generic way for its further use by the different robot skills. In order to facilitate its population to the robot skills, the tool stores robot's poses, and offers ROS2 services to export the information both in XML and YAML formats.

This approach offers a simple and intuitive mechanism to acquire onsite robot data for its further use in the different disassembly tasks. The initial prototype has been implemented and tested on a UR16e, making use of a 3Dconnexion SpaceMouse 6D mouse/joystick and a tablet. The robot guidance relies on a Twist Controller implemented under the ROS2 Control framework.

4 Flexible Production Modules

4.1 Resource Control Framework

The Resource Control Framework serves as a middleware layer between the high-level orchestration system and the robotic execution environment within the RENÉE architecture. At its core lies the **Resource Handler**, a dedicated ROS2 component that interprets and manages task execution requests issued by the **Process Orchestrator**, as described in D4.1.

Upon receiving a task, the Resource Handler initiates the appropriate robotic behaviour by constructing and executing a **Behaviour Tree (BT)** using the `BehaviorTree.CPP` library (*BehaviorTree.CPP*). This structure allows for flexible sequencing, fallback, and parallelization of actions, which are essential for handling complex and dynamic operations in remanufacturing scenarios.

In addition to executing tasks, the Resource Handler is responsible for publishing the state of the robotic resource to the **Workcell-level Digital Twin**, enabling real-time synchronisation and traceability across the system. State information includes execution status, fault flags, and sensor-derived metrics that are critical for system monitoring and higher-level decision-making.

Communication with the physical robotic system is facilitated through the ROS2 Hardware Interface and the ros2_control protocol (*Ros-Controls/Ros2_control*, 2017/2025) This interface layer ensures standardised and low-latency interaction with the robot's actuators, sensors, and controllers, regardless of the specific robotic platform used in each use case.

This modular architecture enables easy reconfiguration and extension of the control framework across multiple pilots, while supporting transparency and interoperability in distributed deployments.

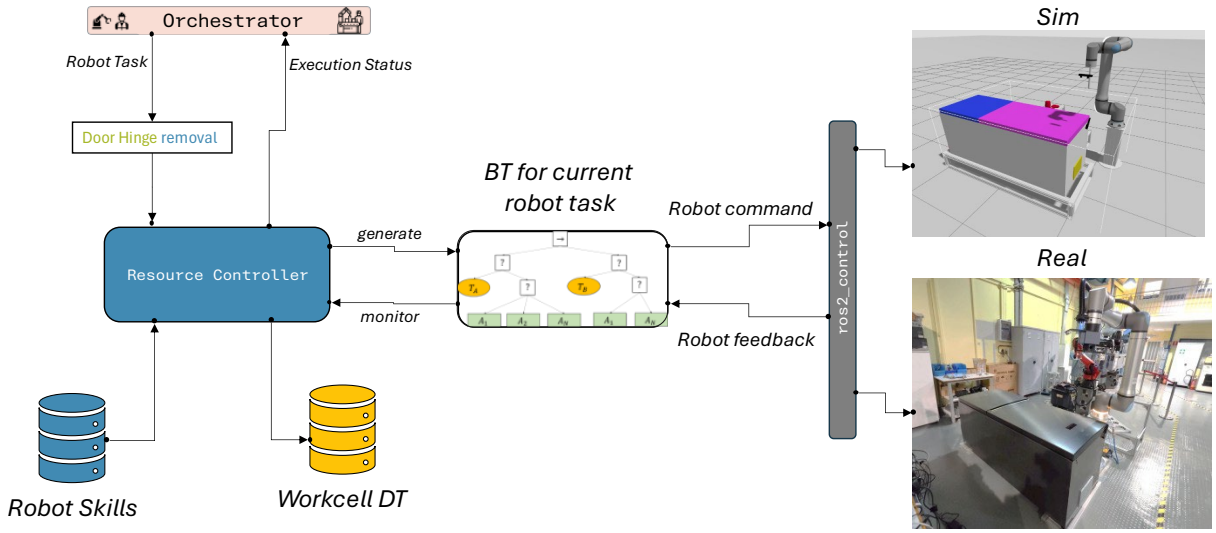

Figure 6: Resource Control Framework.

4.2 Hardware Abstraction Layer

Hardware integration is a key topic in the deployment of new robotic technologies in the shopfloors. Despite the huge advances in AI-based robotic technologies of the last years, the transition to the factories is still an open issue as often research developments and production lines do not rely on the same hardware equipment and interfaces. While ROS2 is the standard development platform in robotics research, the industrial setups are based on PLCs and industrial communication protocols. Due to the industrial nature of the RENÉE use cases, this topic is highly relevant for a successful deployment of the project technologies to real shopfloors.

Therefore, RENÉE proposes the development of a 4.2 Hardware Abstraction Layer (HAL) to link ROS2 to PLCs, focusing on two main tasks, robot control and IO (Input/Output) management. Specifically, the development is based on the implementation of ROS2 Hardware Interfaces as C++ plugins under the ROS2 Control framework for communication with Beckhoff PLCs. Beckhoff PLCs, with their TwinCAT control technology, offer an open ecosystem that facilitates the integration with external devices and technologies through their ADS (Automation Device Specification) protocol,

allowing the PLC variable reading/writing using C++ or Python libraries. This feature is essential for the integration of ROS2 Control framework with Beckhoff PLCs, creating a reliable bridge between both ecosystems.

Specifically, two different ROS2 Control Hardware Interfaces have been developed:

- **Beckhoff Robot HW (Hardware) Interface:** C++ plugin implementing a ROS2 Hardware Interface for robot joint control. It includes joint handles for sending joint positions in high-speed control loops, receiving joint position and velocities as robot feedback. The plugin includes AdsLib, a C++ library to make use of the ADS technology of Beckhoff TwinCAT.
- **Beckhoff IO HW Interface:** C++ plugin implementing a ROS2 Hardware Interface for the management of digital input and outputs. It allows to read/write digital outputs, as well as consulting the state of the digital inputs of Beckhoff IO devices through the plugin's handles. The plugin also includes AdsLib library to interact with the Beckhoff PLC.

The schema shown in Figure 7 depicts the presented developments within ROS2 Control framework.

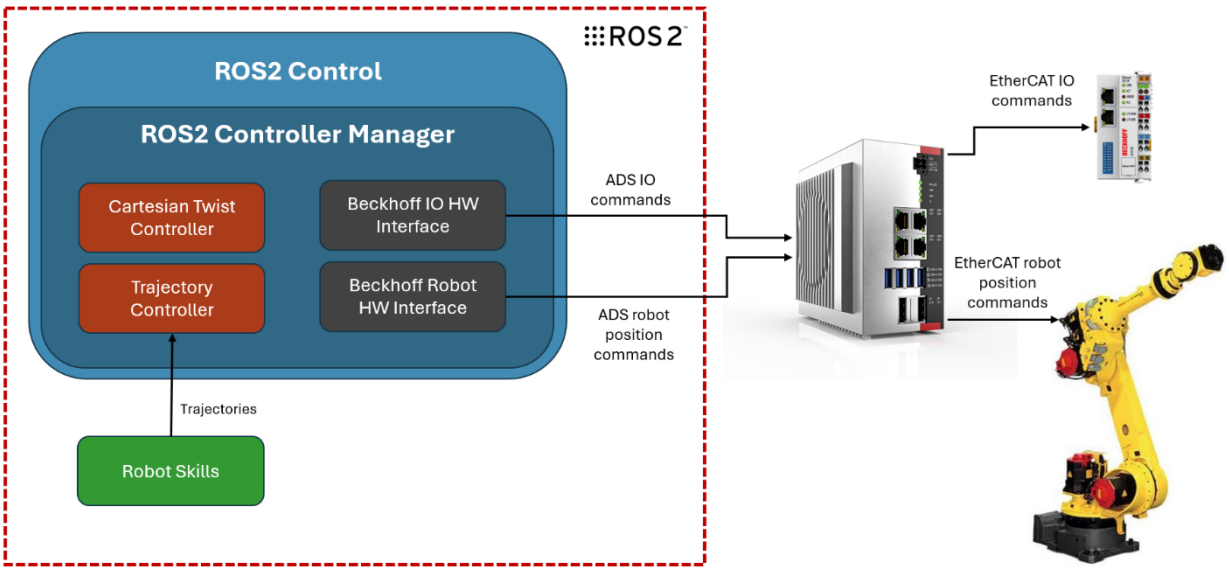

Figure 7: Developed ROS2 HW Interfaces

The presented developments aim to link the ROS2 developments carried out within the RENÉE project with the industrial setups present in the different use cases, facilitating the transition from research environments to industrial settings through the use of PLCs as a key element.

4.3 Robotic Resources Requirements and Selection

4.3.1 Industrial Pilot: Electric motors

One of the most important points on the electric motor disassembly in the EMOTORS use case is the high weight of the motor and its impact on the ergonomics of operators. Additionally, the high forces required for the removal of several parts such as the rotor, pose a scenario where operators need to

interact with robots and additional tools like presses to fulfil the disassembly task. The next table (Table 5) summarizes the requirements of the collaborative electric motor disassembly scenario:

Requirements Table		
Robot Working Area	$20m^2$	
Weight of tools & parts to be handled	**Tool/Part**	**Weight**
	Whole electric motor	*60kg*
	Rotor	*15 kg*
	Stator	*15kg*
	Housing	*2.5kg*
	Inverter	
	Whole motor gripper	*8kg*
	Inverter gripper (external)	
	Screwdriver	*3kg*
Collaborative Functionalities	Co-existence, Cooperation, Collaboration	
Required Communication Protocols	EtherCAT, ROS Drivers	
Special Features	Vision, low-level robot control	

Table 5: EMOTORS Hardware requirements

Based on the previously listed requirements and considerations, a dual robot setup is proposed for the EMOTORS use case. The first robot is a FANUC M-710ic70 industrial robot with a payload of 70kg and a reach of 2050mm that will handle the complete electric motor, as well as the heavy parts. Additionally, a second robot will handle the unscrewing and small part removal, specifically a UR16e robot with 16kg of payload.

To facilitate the development and integration of components and technologies, the prototype will be deployed at TECNALIA. This deployment is divided into two phases, the initial prototype (M18) where all the hardware components are installed and the second prototype (M36) with the software component integrated into the cell:

Figure 8: EMOTORS testbed phase 1

a) **Phase 1:** In this initial phase, the main hardware elements of the cell have been installed and fabricated, specifically:
    a. FANUC M-710ic70 with the whole motor gripper.

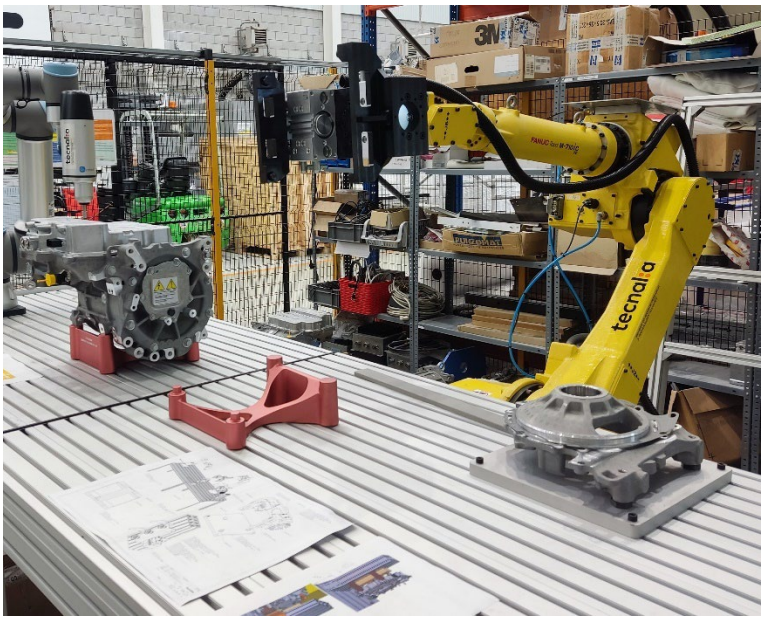

Figure 9: EMOTORS testbed - Industrial high-payload robot for whole motor manipulation

    b. UR16e robot with OnRobot screwdriver.

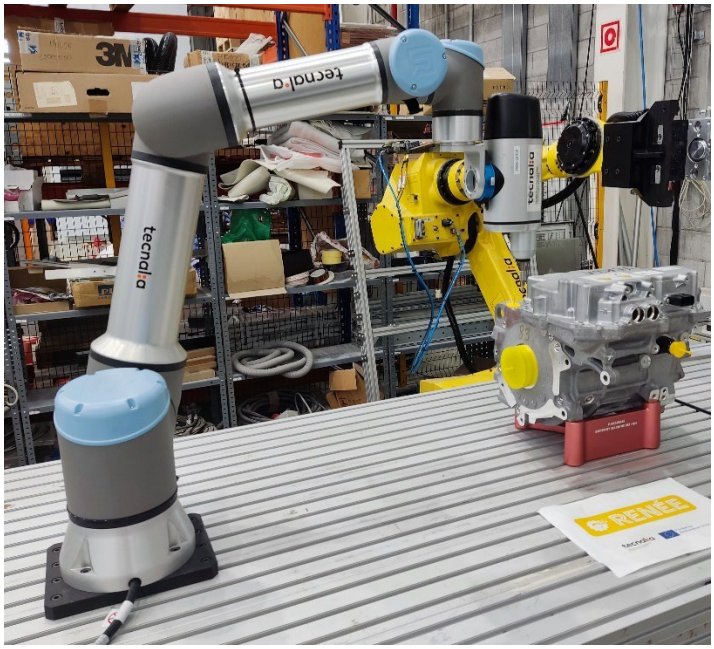

Figure 10: EMOTORS testbed -Collaborative robot for unscrewing operations

c. Inverter removal external suction gripper.

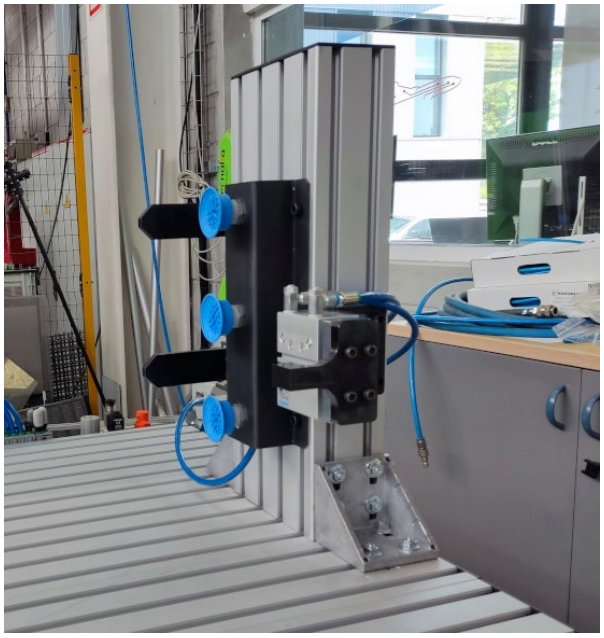

Figure 11: EMOTORS testbed- External inverter removal suction gripper

d. Finally, the main electric and pneumatic connections between robots and grippers have been carried out. The setup is completed with two aluminium profile tables of 2x1m each, offering a total co-working area of 4x1m.

b) **Phase 2:** During the second phase, the software framework will be set up, integrating and developing the modules to manage the robots, grippers and additional devices in ROS2. This software framework will make use of a Beckhoff PLC to interface the industrial elements that currently have no ROS2 driver.

4.3.2 Industrial pilot: Household Appliances

Considering the disassembly of the refrigerator in the ARCELIK use case, the variation of the core components (in terms of size, weight, geometry, etc.), among the various models, need to be extracted from the refrigerator frame poses significant limitations and constraints to the selection of the appropriate resources to handle the disassembly process (Table 6). Thus, the combination of a human operator and a cobot has been selected as the optimal approach. The capabilities and limitations of both these resources are taken into considerations towards optimally allocating the disassembly tasks among them (T4.1 – Process Scheduler and Orchestrator, PSO).

Requirements Table		
Robot Working Area	$20m^2$	
Weight of tools & parts to be handled	**Tool/Part**	**Weight**
	Refrigerator Frame	*50-80kg*
	Compressor	*3-23 kg*
	Doors	*5-25kg*

	PCB	*0,5kg*
	Mechanical Components (hinges, plastic covers, etc.)	*0,1-0,5kg*
	Gripper	*12kg*
	Screwdriver	*1kg*
Collaborative Functionalities	Co-existence, Cooperation, Collaboration	
Required Communication Protocols	Profinet, ROS Drivers	
Special Features	Force control	

Table 6: ARCELIK Hardware requirements

The most important constraint is for the robot to effectively handle the manipulation of heavy components (the refrigerator doors) to alleviate the ergonomic stress from the operator. Thus, the robot should have the appropriate payload to handle these components while also allowing safe collaboration with the human. For these reasons, the UR20 solution has been selected, combining collaborative functionalities with significant payload capacity.

To effectively handle the integration of all the software components, the testbed implementation has been split into two phases, an initial prototype testbed at LMS facilities and the final integrated testbed at the facilities of TF – CC.

a) **Phase 1**

At the initial phase of implementation, only the human operator and the UR20 are considered targeting the disassembly of the doors, the PCB and the mechanical components as well as the shelves and the fan from the interior of the refrigerator (Figure 12).

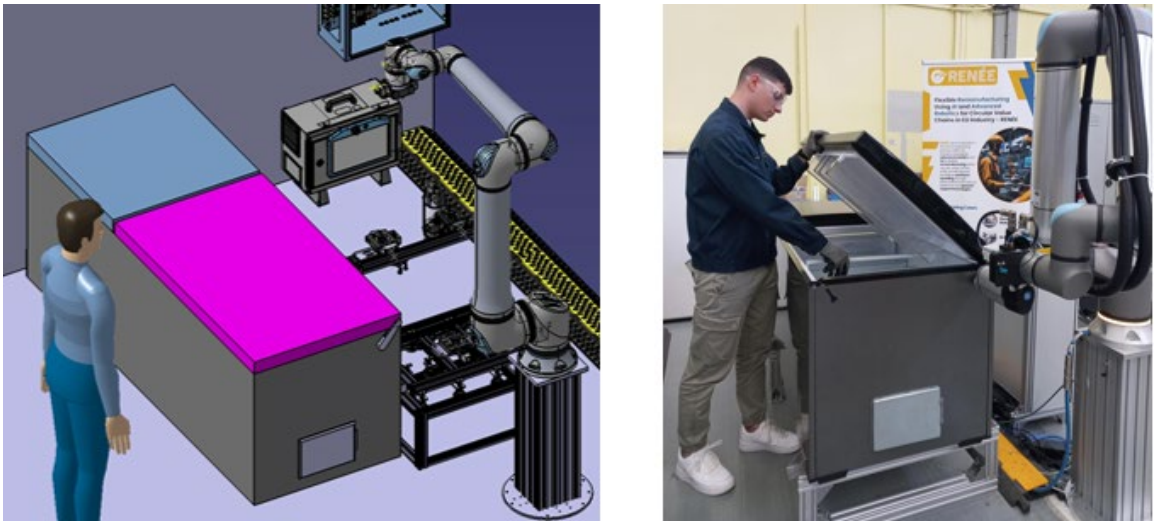

Figure 12: ARCELIK testbed phase 1

b) **Phase 2**

During the second phase, the final prototype is expected to handle the compressor as well as manipulating the whole refrigerator frame. These components exceed the payload capacity of the UR20 and thus the introduction of an additional heavy payload robot is planned for the manipulation of the heavy components, while the human operator and the

light payload cobot will handle the disassembly of the rest of the components as performed in the initial prototype of the testbed (Figure 13).

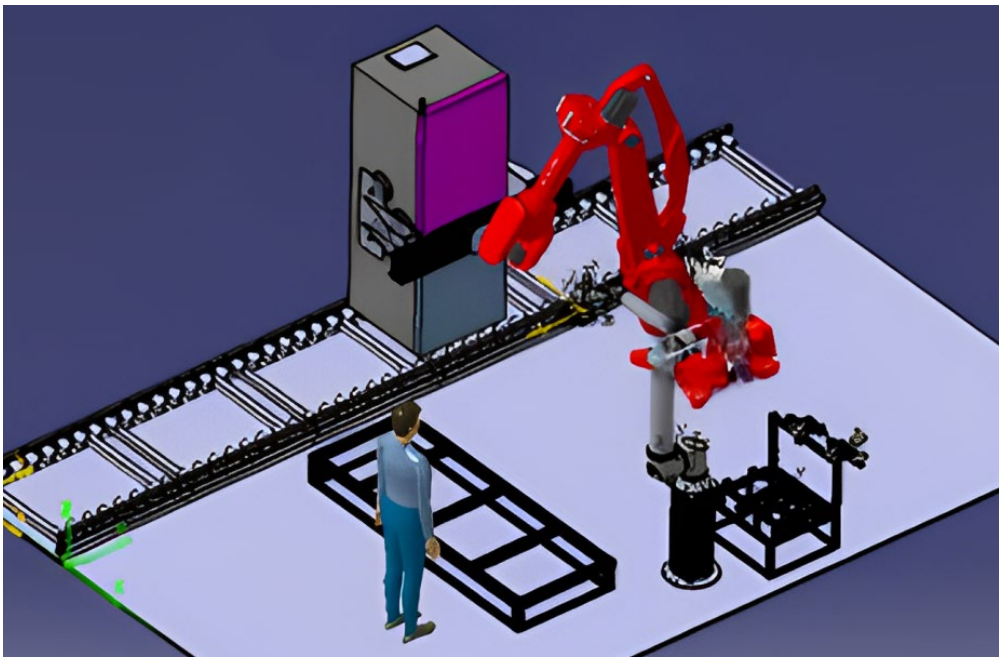

Figure 13: ARCELIK testbed phase 2

4.3.3 Industrial pilot: Robots

In the CAMPETELLA use case, addressing the disassembly of industrial robots presents a challenge due to the variability of core components, in terms of size, weight, and geometry, within a specific robot model. To effectively manage this, it is crucial to extract these component details from the specific robot model (as referenced in Table 7) to inform the selection of appropriate disassembly resources.

Requirements Table		
Robot Working Area	$12m^2$	
Weight of tools & parts to be handled (some parts as samples)	**Tool/Part**	**Weight**
	Rack	*3 kg*
	Axis fixing plate	*2 kg*
	Belt tensioning plate	*1.8 kg*
	Cable chain	*1.5-3.5 kg*
	Sliding rails	*1.8 kg*
	Mechanical Components	*0,1-0,5kg*
	Screwdriver	*1kg*
Collaborative Functionalities	Co-existence, Cooperation, Collaboration	
Required Communication Protocols	ROS Drivers	
Special Features	Force control	

Table 7: CAMPATELLA hardware requirements

The chosen strategy for this pilot involves a collaborative effort between a human operator and a cobot. This decision stems from the need to optimally allocate disassembly tasks (T4.1 – PSO) by carefully considering the distinct capabilities and limitations of both human and robotic resources. A key objective is to mitigate ergonomic stress on the human operator. Consequently, the cobot is tasked with manipulating heavy components, such as motors, long guiding rails, and support fixing plates, and executing complex dismounting procedures for certain parts.

The selection of the UR5e cobot solution, as shown in Figure 14, configured as a mobile manipulator to expand the workspace of the arm and enable access to different parts of the product, was driven by these requirements. This solution offers the necessary payload capacity and dexterity to handle the challenging components and their intricate disassembly processes, while ensuring safe collaboration with the human operator.

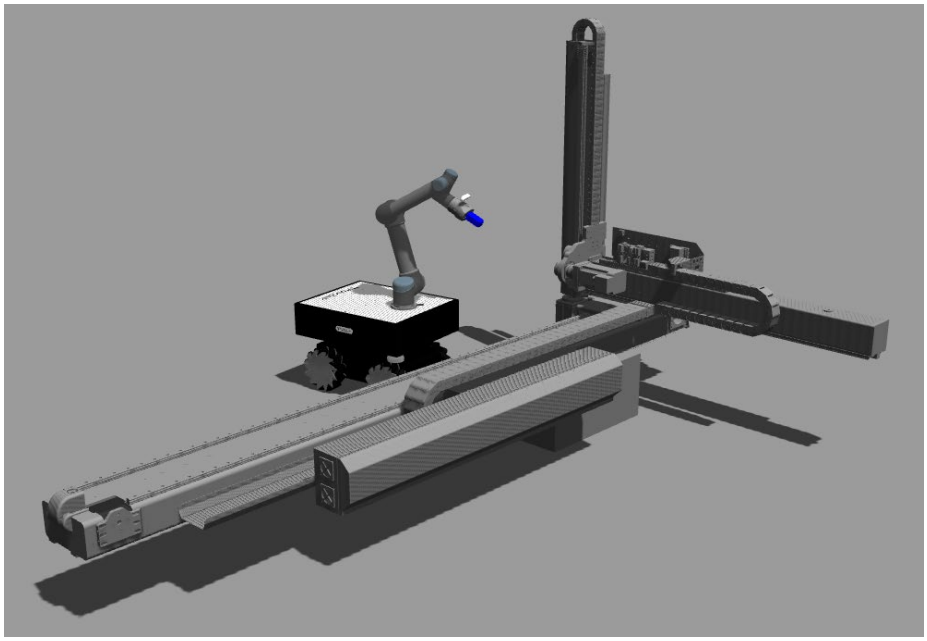

Figure 14: CAMPETELLA testbed - Mobile manipulator solution for the CRC pilot

To ensure the seamless integration of all software components, the testbed implementation will be conducted in two distinct stages:

1. **Initial Prototype Testbed:** This first stage will involve establishing a preliminary testbed at the IIT facilities. Figure 15 represents the layout of the initial testbed.

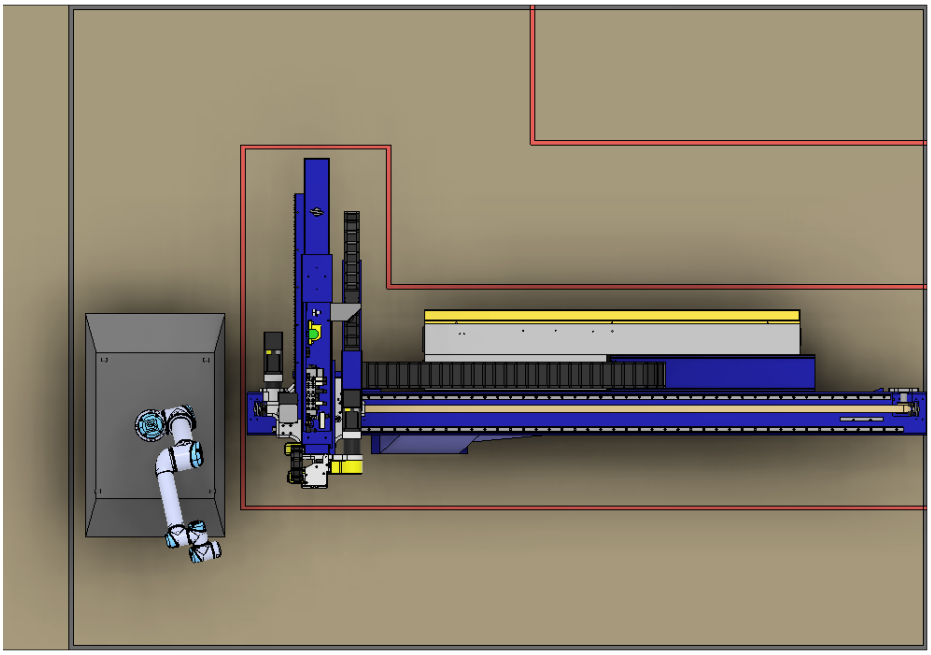

Figure 15: CAMPETELLA testbed layout

2. **Final Integrated Testbed:** Subsequently, the final, fully integrated testbed will be developed and deployed at CAMPETELLA's facilities.

The demonstration of the system will also proceed in two phases:

c) Phase 1: Hardware Installation and Core Functionality
In this initial phase, the primary hardware elements of the robotic cell have been installed and are operational. These include:
- Mobile manipulator
- Vision modules
- CAMPETELLA Cartesian robot
- Safety environment facilities for robotic cell
- The core functionalities of the robotic cell, such as navigation, part manipulation, cleaning operations, and part inspections, are also established during this phase.

d) Phase 2: Software Framework Integration and Development
During the second phase, the software framework will be configured. This involves integrating and developing the necessary modules in ROS2 to manage the robots, grippers, and other auxiliary devices.

4.3.4 Industrial pilot: Bicycles

The remanufacturing activity within the RENÉE use case will focus on assessing the condition of three main components: brakes, forks, and carbon fibre parts. The robotic activity will specifically target the inspection of carbon fibre components through non-destructive testing techniques, with a particular focus on active thermography. This technique involves heating the component with a thermal source, such as a halogen lamp or flash, while a thermal camera captures infrared scans.

To facilitate this process, both the thermal camera and the heat source will be mounted on a robot. This setup enables precise positioning close to the carbon frame and allows for movement along the frame tubes to perform the thermographic inspection. The total weight of the thermal camera and lighting system was estimated, and based on this payload, the Comau Racer 5 robot was selected as it can adequately support the required load. The robotic station will be installed at a Decathlon store and will consist of the robot and a linear axis to extend the arm's reach, increasing the operational flexibility of the robotic cell.

The use of a robotic system offers significant advantages. A manual inspection that is both repeatable and certifiable would require a qualified operator with at least a Level 1 certification. In contrast, a properly calibrated and potentially certifiable robotic system can deliver consistent, repeatable, and certifiable results, providing added value for the end customer.

Currently, several operational setups are being evaluated, depending on the specific characteristics of the thermal camera and heating sources. One possible configuration involves the robot moving along the frame to enable close-range scanning by the thermal camera. Alternatively, the camera and lights could be kept at a fixed distance from the surface, with the ability to approach specific areas of interest if anomalies are detected or if further local analysis is needed. To enable this movement, STAM will develop a machine vision system capable of identifying the geometry of the bike frame to compute the trajectories for the thermographic tool.

The robot will operate in cooperation with a human operator. While the robot performs the thermographic inspection of the carbon frame, the operator will be nearby, monitoring the real-time output of the analysis algorithms.

Requirements Table		
Robot Working Area	$20m^2$	
Weight of tools & parts to be handled	**Tool/Part**	**Weight**
	Thermal camera	*0.3 Kg*
	Heat source	*2 Kg*
	Thermal system holder	*0.5 Kg*
Collaborative Functionalities	Co-existence, Cooperation	
Required Communication Protocols	ROS drivers	
Special Features		

Table 8: DECTHLON Hardware Requirements

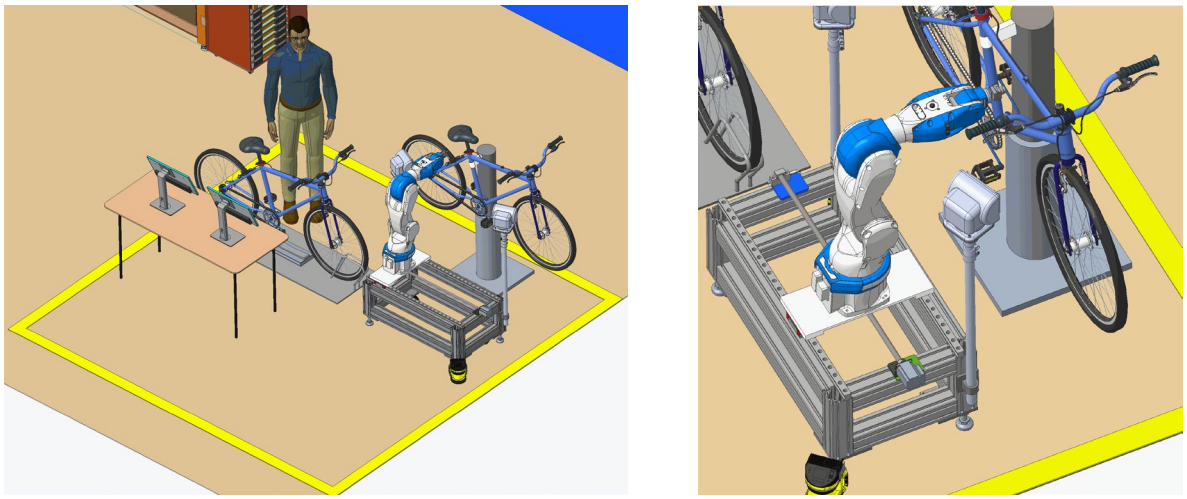

Figure 16: DECATHLON Testbed

5 AI-Enhanced Perception System

5.1 Flexible Robot Perception Framework

This section outlines the initial developments of the AI-enhanced perception system designed to support robotic operations in flexible remanufacturing environments. The system integrates data-driven *object detection*, *semantic understanding*, and *scene reconstruction* capabilities to enable robust perception under open-world and variable conditions. By leveraging deep learning models and adaptive sensor fusion strategies, the perception framework provides the foundational layer for object localization, status inference, and decision-making in human-robot collaborative scenarios.

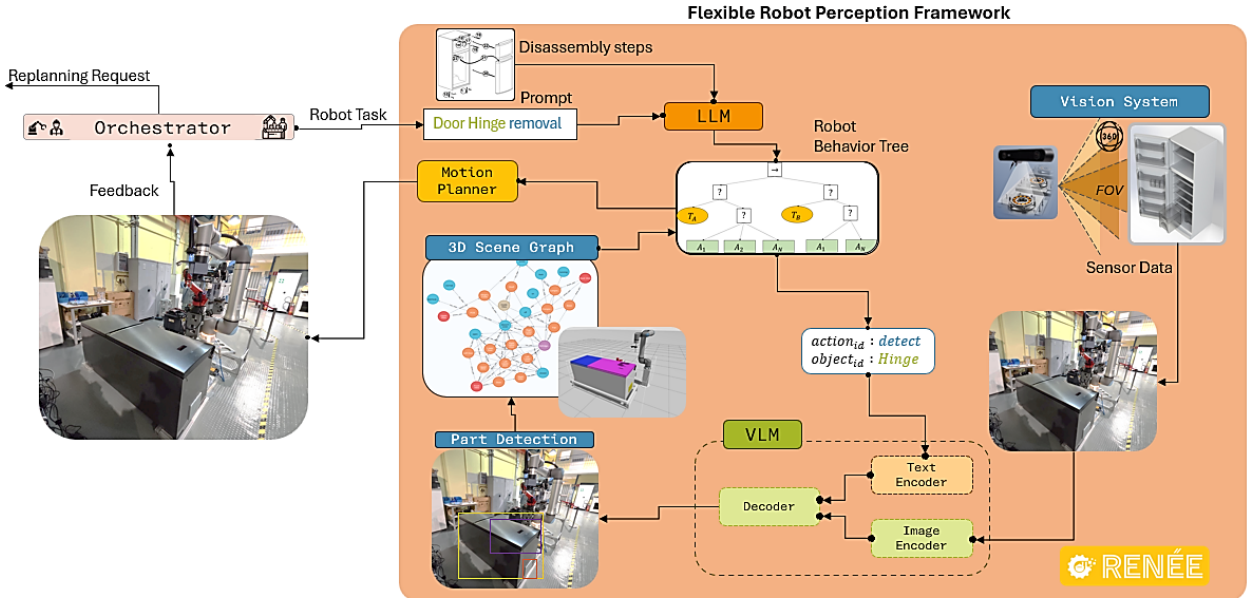

Figure 17: Flexible Perception Framework Architecture

5.2 Perception Modules

In each case, different perception modules have been developed in order to facilitate autonomous remanufacturing processes with robots. Below are presented the prototypes of the perception modules.

5.2.1 Multiple Parts Detection

This perception module is designed to recognise and localise various product components within cluttered or semi-structured environments. This supports the remanufacturing processes execution by the autonomous robotic resources. The module relies on deep CNNs object detection models, which has been trained on a custom dataset for each product.

The training procedure is conducted locally, ensuring data privacy. After training, the best-performing model checkpoint is stored on the RENÉE platform, enabling access for inference or further training.

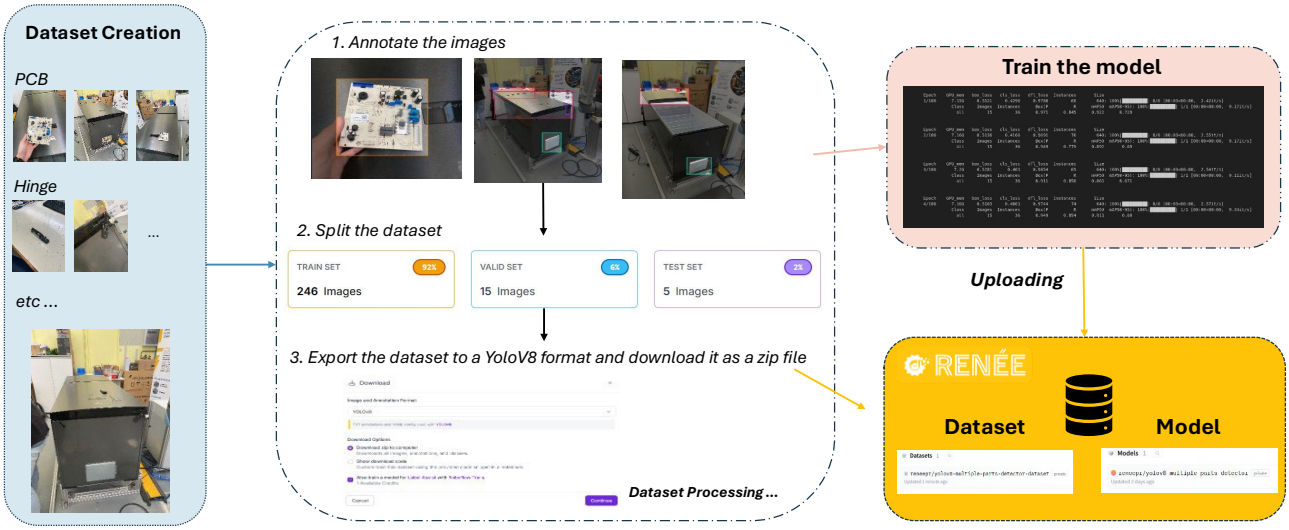

Figure 18: Architecture Diagram for Multiple Parts Detection Module

The Multiple Parts Detection module has been tested in the ARCELIK and CAMPETELLA use cases, demonstrating high performance in accurately identifying key components. In ARCELIK, the module effectively detected various refrigerator parts with high confidence, validating its applicability for remanufacturing scenarios. For CAMPETELLA, the module is crucial for recognizing and localizing components and screws of industrial robots, guiding robotic actions for disassembly, unscrewing, and pick-and-place operations. In the EMOTORS case, this AI-enhanced perception system is applied during the initial disassembly stages of electric motors. It utilizes object detection models to recognize and localize connectors and auxiliary components in cluttered or semi-structured environments, thereby supporting high-level planning and task allocation for disassembly. The same module will be trained and tested on the remaining RENÉE use case. In the DECATHLON use case, the system will identify the geometry and position of the bicycle frame within the inspection cell, enabling the robot to accurately perform in-depth inspections with a thermal camera.

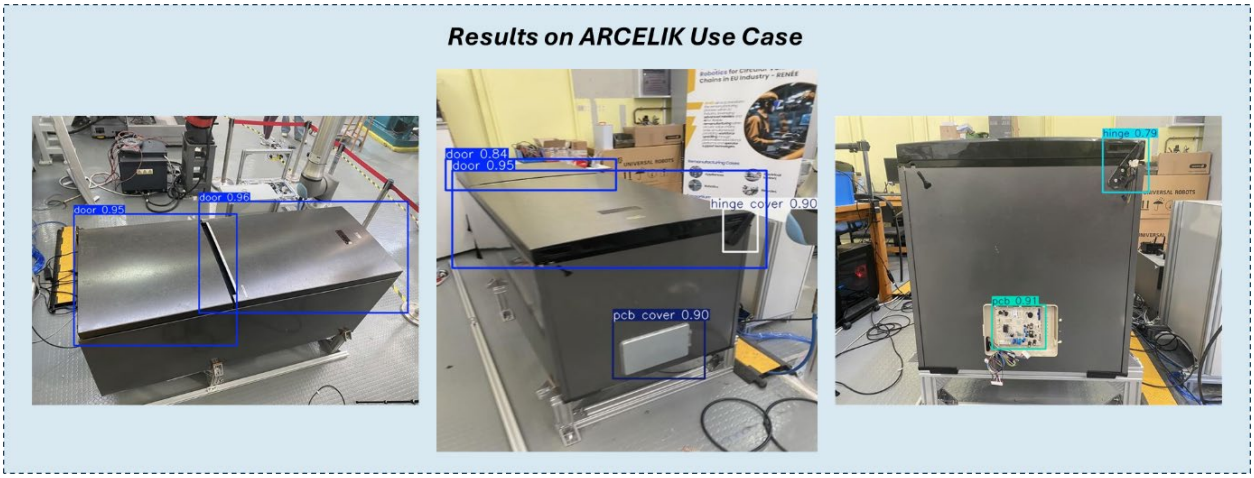

Figure 19: Multiple Parts Detection Module tested on ARCELIK use case

Figure 20: Multiple Parts Detection Module tested on CAMPETELLA use case

5.2.2 Screw Detection and tracking

Robotic unscrewing of EoL products presents numerous challenges related to products diversity, wear, design and variability in assembly requirements. An automated unscrewing system is called to handle various screw sizes and types. The fasteners are very often rusted, deformed, contaminated or tightened with unexpected torque. Other potential issues are the fastener's localization & accessibility.

A robust robotic unscrewing system is required to be equipped with perception systems that could identify these uncertainties and provide all the required information for the execution of the disassembly process. Before unscrewing, the system should identify the type of screw type (Philps, Hex, Torx), the screw head size, the screw location, as well as the screw health condition. However, even the above data are successfully calculated, parameters like screw length or thread hole condition could not specified without removal. For this reason, a perception module is crucial during the unscrewing to ensure successful execution.

This perception module, leveraging both visual feedback and force-torque sensor data, is being developed to support accurate identification and alignment of screw positions, enabling automated or assisted fastening/unfastening.

Visual Perception

Automating robotic screwing processes requires precise 3D pose detection and tracking of the screw or bolt head. This is achieved through a perception pipeline within the screwing perception module. Initially, the visual modality employs a CNN model, trained on screw images, to detect the screw's initial 2D position and head type (Figure 21, Figure 22). A 3D sensor, mounted on the robot arm, simultaneously provides both 2D and depth images. The 3D pose of the screw head, critical for robot movement, is then extracted from the depth sensor's point cloud. A synchronization algorithm aligns the point cloud with the 2D image, enabling the estimation of the depth value directly from the detected 2D point (Figure 23).

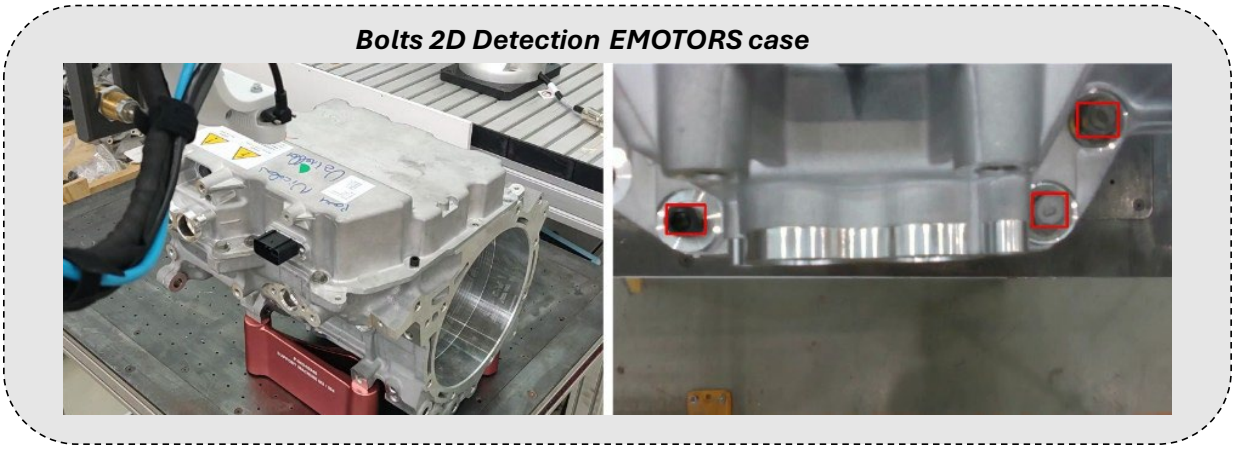

Figure 21: Bolt detection on electric motor

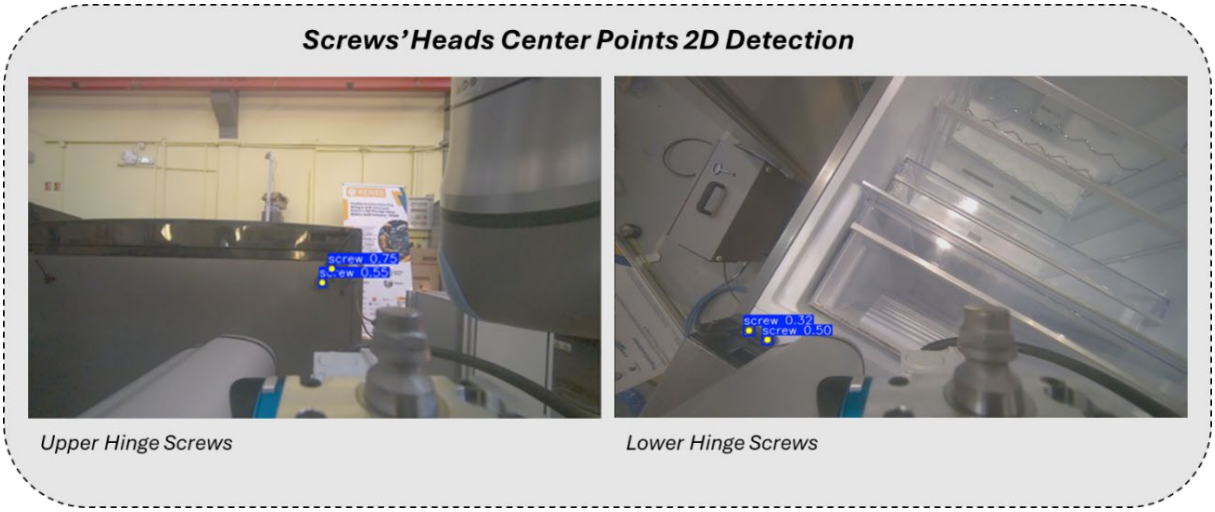

Figure 22: Screws detection refrigerator

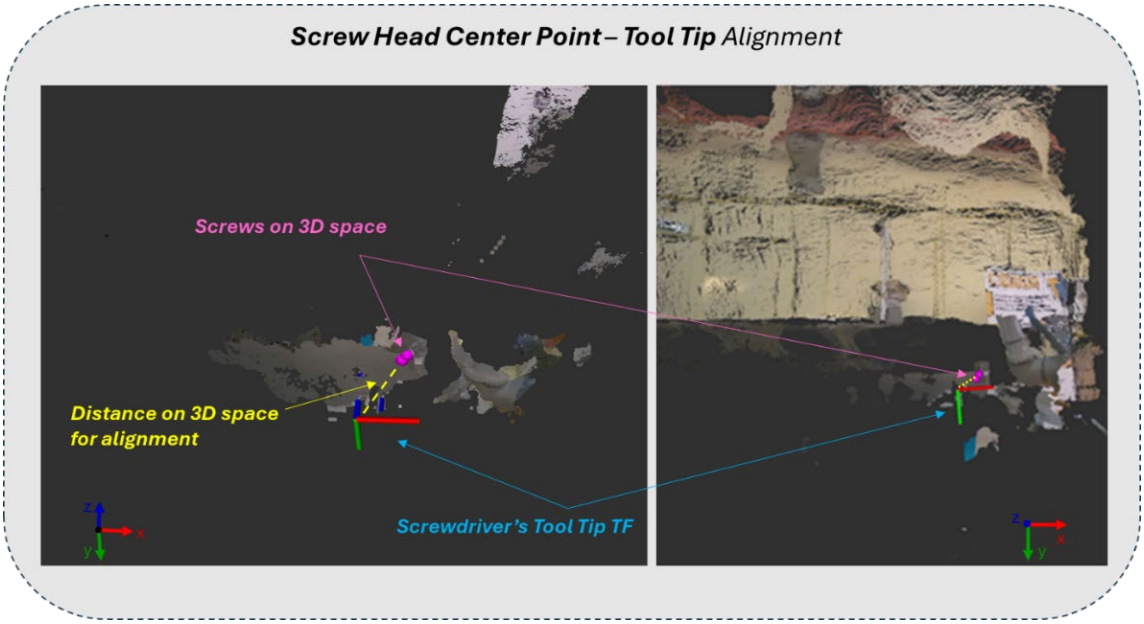

Figure 23: Screw 3D pose estimation

The camera is rigidly mounted on the robot arm alongside the robotic screwdriver. The pre-calibrated, static transformation (TF) between the camera and the screwdriver's tool tip ensures accurate localization of the tool tip within the common 3D reference frame. With the screw head and tool tip positions precisely known in 3D space, the robot guides the tool tip into alignment with each screw. Once alignment is achieved, the unscrewing process begins.

Force - Torque Perception

To verify physical contact and monitor task execution, force-torque feedback from the screwdriver will be continuously analysed. This data will be used to confirm that the tool will have engaged with the screw and will be applying the necessary force for unscrewing. A continuous analogue force signal, over time, will indicate that the unscrewing process is proceeding as expected. Moreover, by monitoring the force-torque data, the system will be able to detect anomalies or task failures. Such failures can be cam-out (slipping between the screwdriver and the screw head), misalignment between the tool and the fastener, worn or stripped screw heads. In such cases, the module will trigger a notification to the human operator or planner, mark the task as incomplete, and assign the task to the operator.

This perception-driven screwing module has been tested within the ARCELIK and EMOTORS use case, using a ZED X Mini stereo camera mounted on the robot and the OnRobot screwdriver with embedded force-torque sensor.

These modules are being iteratively developed and tested in both simulated and physical setups, with a focus on robustness, real-time performance, and seamless integration into the flexible production module workflow.

5.2.3 Operator's Action Recognition

This perception module, based on Vision-Language Models (VLMs) and Large-Language Models (LLMs), is being developed to enable understanding of human actions during collaborative tasks. By grounding visual observations in natural language descriptions, this module supports intent recognition, task progress monitoring, and adaptive robot behaviour in shared workspaces. The module is designed to extract a structured list of tasks performed by analysing the process in real time. The approach integrates multiple inputs consisting of visual, audio and speech. The module's architecture is shown in Figure 24.

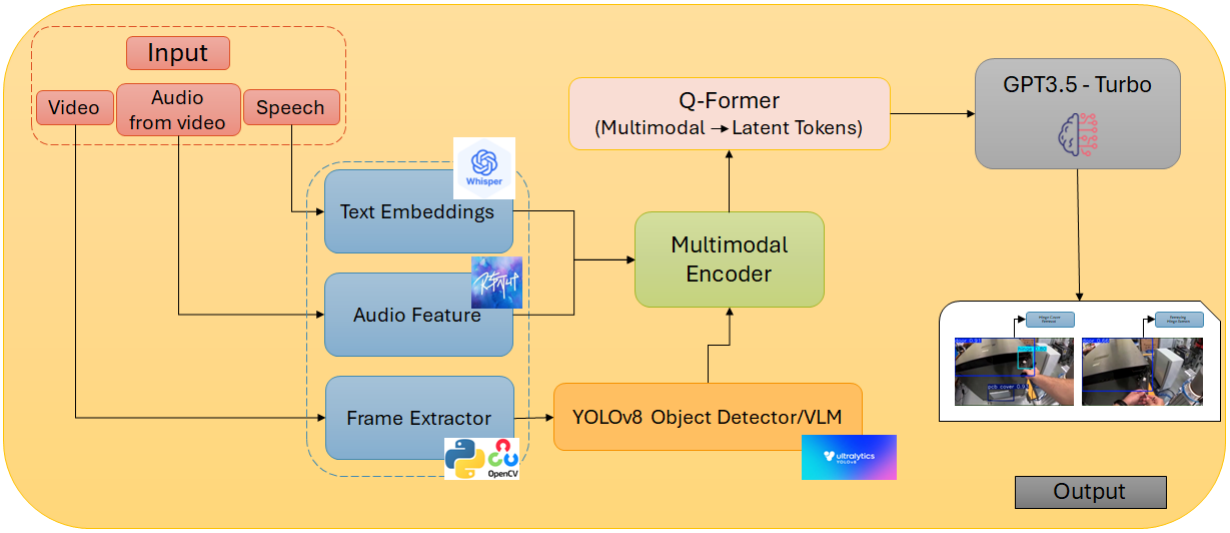

Figure 24: Action Recognition Module Architecture

Both visual execution of tasks and the operator's live verbal explanations are parsed and processed. Speech is transcribed using a speech recognition model, specifically the Whisper-large-V3 model from OpenAI, and converted into pure text. In parallel, non-verbal audio from the live stream is processed through an audio encoder to extract features. The model used is the HTS Audio-Transformer. Simultaneously, visual frames are extracted in real time using a computer visual toolchain, the python OpenCV.

The extracted data are integrated into a single contextual vector by a Multimodal Encoder. The visual aspect is also enriched by Multiple Parts Detection module to accurately identify the parts detected in every frame. To retain essential contextual signals, a Q-Former module converts the multimodal vector into latent tokens, suitable for interpretation by an LLM.

These tokens are passed to the LLM, specifically GPT 3.5-turbo, which processes the context and outputs dynamically updated list of tasks. The result is a structured JSON file containing the tasks performed in order of occurrence, in natural language. This output enables live monitoring of the process and potentially dynamic rescheduling. The prototype was tested in the ARCELIK use case, on a sample video of Refrigerator Disassembly, where both operator and robot working on the same process. In Figure 25, is shown two sample frames of the video from the ARCELIK use case, with two different human tasks being performed.

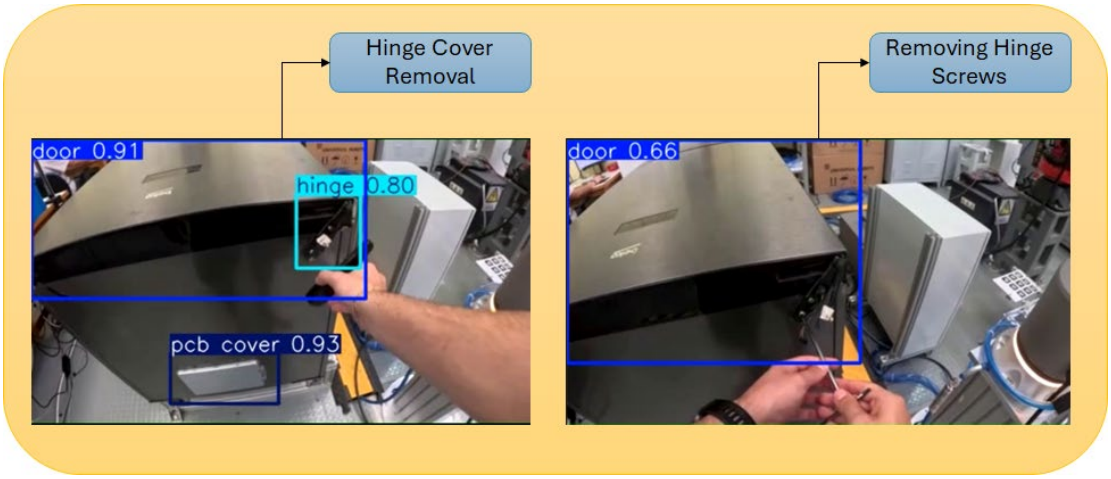

Figure 25: Captured Human Tasks

5.2.4 Product Visual Assessment

This perception module focuses on evaluating both the external and internal visual condition of refrigerator units and provide input to the Product State Diagnosis (PSD) module, presented in deliverable D3.1, for estimating the state of the refrigerator allowing to adjust the remanufacturing strategy accordingly.

This perception component provides semantic information regarding surface wear, contamination, or mechanical damage, which is essential for decision-making processes. In particular, the module assesses the condition of refrigerator components using a CNN object detection model, trained locally.

The objective of this module is to automatically detect visual anomalies such as dents, cracks, scratches, and dirt accumulation. These visual cues are then used to estimate the overall condition of the refrigerator and its suitability for reuse or further remanufacturing operations.

To train the model effectively, it was necessary to construct a representative dataset. Images including cracks, dents, scratches, and dirt on a variety of materials and surfaces were collected to maximize generalization. The data preparation pipeline involved the following steps:

- Image collection from diverse sources and materials (Figure 26).

Figure 26: Images Collected for the Dataset

Annotation of relevant visual defects (Figure 27).

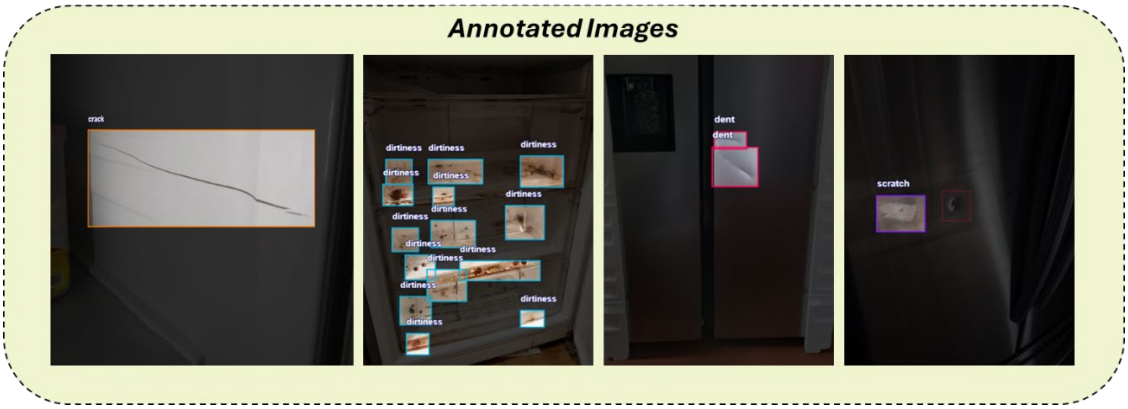

Figure 27: Annotate the images to prepare the dataset for testing

- Formatting the dataset to comply with YOLOv8 requirements (Redmon et al., 2016).
- Uploading the dataset to a centralized database.
- Accessing and utilizing the dataset and trained model from this database.

Training and inference were executed locally to ensure data security. The dataset and trained model are stored on a private database.

Preliminary results for the ARCELIK use case suggest promising potential for the proposed approach. However, current performance is limited by the relatively small size and generality of the training dataset. Initial tests in refrigerator units' images have demonstrated suboptimal accuracy, likely due to domain-specific conditions and visual features unique to refrigerator surfaces.

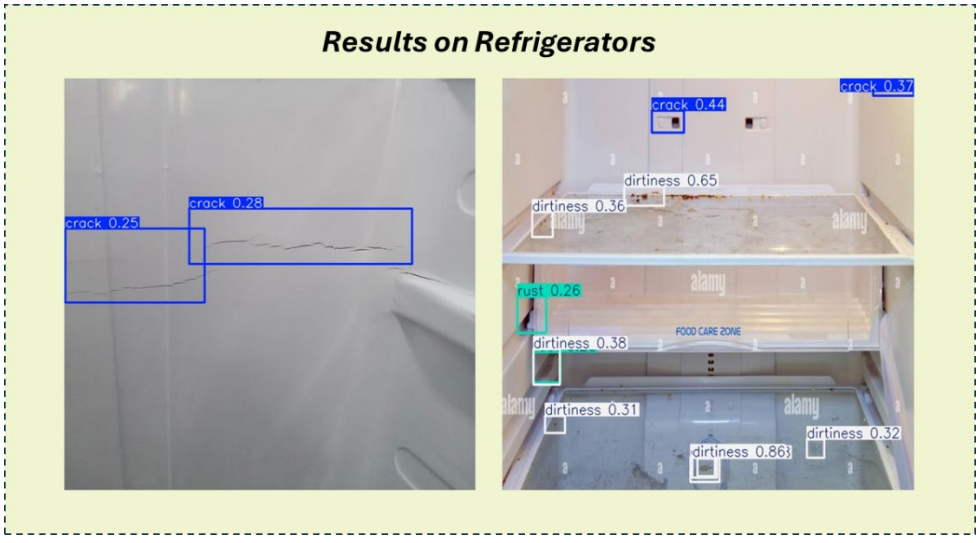

Figure 28: Visual Condition Assessment Module tested on refrigerator images

To address this fundamental limitation, generative AI techniques will be employed to synthetically augment training datasets, thereby improving model robustness and generalisation capabilities across diverse defect types and component conditions. In particular, the COIGAN (Controllable

Object Inpainting with Generative Adversarial Networks) framework addresses the critical challenge of defect detection training data scarcity in cartesian robot remanufacturing applications (Biancucci, Massimiliano, 2024/2025). This innovative approach generates synthetic defects on component surfaces through user-defined drawable defect maps, enabling robust training of visual inspection algorithms without requiring extensive datasets of real defective parts. The system employs multiple independent defect channels that allow users to specify different defect types and locations through intuitive mask drawing interfaces, which are subsequently transformed into photorealistic synthetic anomalies on clean component surfaces. This controllable defect synthesis capability ensures comprehensive coverage of potential failure modes whilst maintaining the flexibility to generate application-specific degradation patterns relevant to cartesian robot component lifecycles.

The integration of COIGAN-generated training data with the eye-in-hand camera system enables more robust defect detection capabilities, improving the overall effectiveness of the automated inspection workflow within the remanufacturing process.

All perception modules will interface with the broader RENÉE digital infrastructure, feeding data into digital twins and digital product passports to support decision-making and traceability throughout the remanufacturing process.

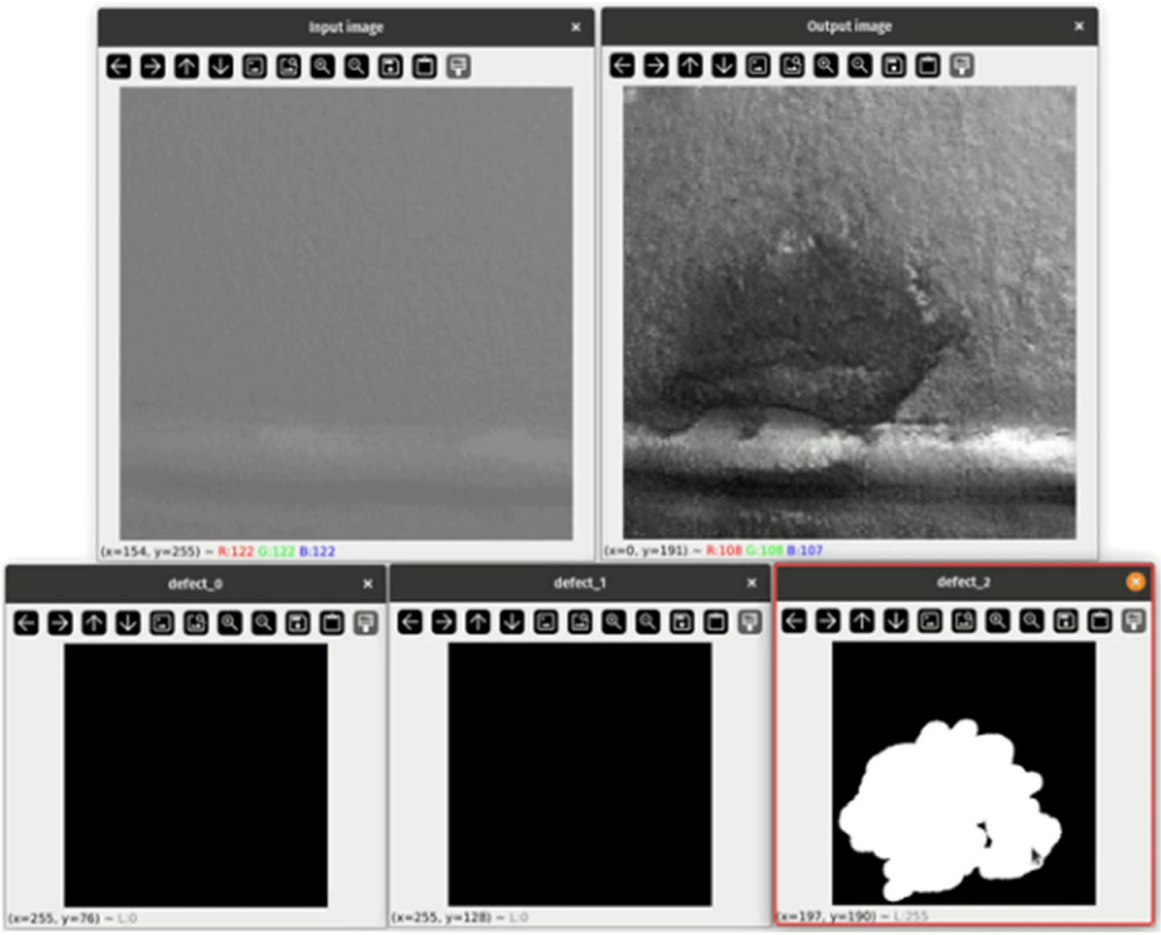

Figure 29: COIGAN interface demonstrating controllable defect generation for cartesian robot component inspection

Subsurface Defect Detection via Thermography

The system shall be capable of inspecting components to detect subsurface defects such as cracks, delamination, or voids. It must accurately acquire thermographic data during the inspection process and process this data to extract meaningful thermal information. The procedure relies on active thermography, a technique that involves heating the component using a thermal excitation source and acquiring thermal maps during both the heating and cooling phases. The presence of a defect is revealed in the thermal scan as a variation in specific parameters such as temperature and in case of periodic heating phase or amplitude. These anomalies indicate possible material discontinuities or subsurface damage within the component. A machine learning algorithm is then applied to the acquired thermal data to automatically identify and classify defect patterns with increased reliability. This module is tested initially in the DECATHLON's use case for detecting subsurface defects in the skeleton of the bicycles.

5.2.5 Perception Modules for Inspection

This section outlines several key perception modules developed to enable automated inspection and characterization of various components and systems. Each module employs distinct sensing modalities and data processing techniques to achieve its specific objectives, providing a foundation for objective evaluation and defect detection.

Mechanical Response Characterization

This module focuses on analysing the mechanical behaviour of components under applied loads to assess their structural integrity and performance characteristics. These modules are used to analyse the suspension forks of bicycles in order to assess their condition and determine whether they are still suitable for use. The developed tool records the force applied by the operator over time while simultaneously measuring fork travel. This enables the generation of characteristic curves for each component, allowing for an objective evaluation process.

Figure 30: Bicycle under fork suspension analysis

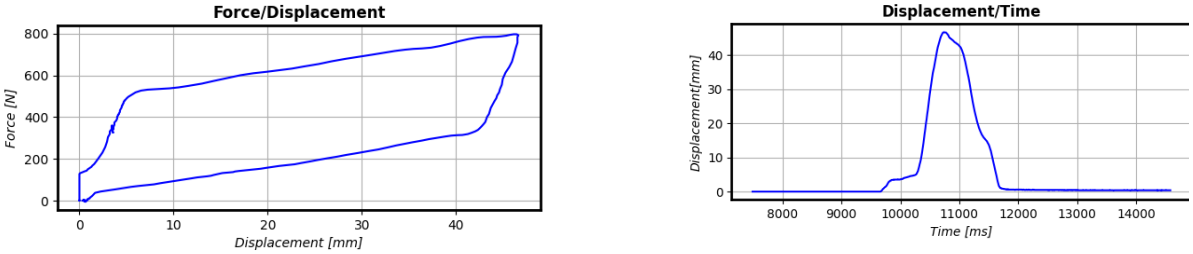

Figure 31: Results of fork suspension analysis

Dynamic System Performance Analysis

Within the RENÉE project, the aim is to develop an automated inspection system inspired by the standard ISO 4210 brake testing procedure, based on a motorized roller bench that simulates braking conditions in a controlled and repeatable manner.

The system shall accurately detect and measure both an applied input force (e.g., to an actuator or lever) and the resulting output force or torque within a dynamic system. It must also record relevant motion parameters, such as rotational speed, to allow for comprehensive performance evaluation against specified criteria. The perception system is designed to ensure accurate and synchronized acquisition of all relevant parameters to assess system linearity and detect anomalies such as inconsistent or insufficient responses.

A test system simulates operational conditions using a motorized mechanism (e.g., a roller bench or test stand) capable of maintaining controlled speeds or applying controlled loads. As an actuation or braking mechanism is engaged, the system measures the resulting force or torque generated. A controlled input force is applied to the mechanism using an automated actuator, ensuring consistent and repeatable operation. This integrated setup enables the evaluation of dynamic performance by verifying the proportional relationship between the applied input force and the resulting output, in compliance with relevant performance standards and requirements.

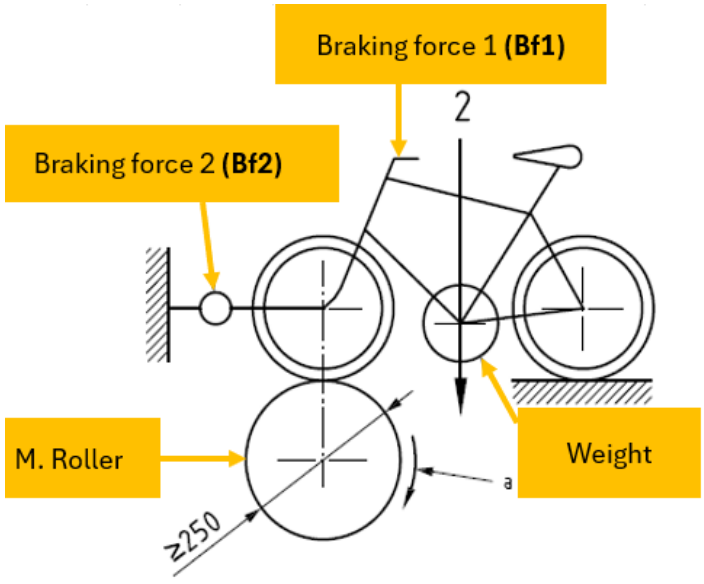

Figure 32: Automated braking inspection system

6 Intelligent End Effectors and Smart Mechatronics

6.1 Design and Development of Intelligent Grippers

Although several of the previous objectives were specifically tailored to fit certain pilot cases of the project, it has always been essential to identify and develop a common solution capable of addressing all pilot cases comprehensively. The four pilot cases involve the remanufacturing of household appliances (refrigerators), the reuse of robotic components, the remanufacturing of electrical motors, and the restoration of bicycle components, in collaboration with ARCELIK, CAMPETELLA, EMOTORS, and DECATHLON, respectively. The products to be remanufactured exhibit a wide range of physical conditions, degradation states, and component configurations, ranging from heavily worn components to partially functional items.

In this diverse remanufacturing scenario, IIT proposed a general technological solution to meet the requirements of the four pilot cases. After a thorough examination of the facilities and remanufacturing processes for the four pilot cases, it was possible to identify macro-categories of grasping requirements common to the four use cases. The results of the investigation of the requirements for the pilot cases are reported in Table 9 and Table 10.

Features	EMOTORS	ARCELIK	CAMPETELLA	DECATHLON
Cleaning	Maybe	No	Yes	Maybe
Inspection	Yes	Yes	Yes	Yes
Unscrewing	Yes	Yes	Yes	No
Component Removal	Yes	Yes	Yes	Yes
Component Placement	Yes	Yes	Yes	Yes
Component Testing Aid	-	-	Maybe	Yes

Table 9: List of features per every use case

Requirements	EMOTORS	ARCELIK	CAMPETELLA	DECATHLON
Screws	Yes	Yes	Yes	Yes
Bih/Heavy Components	No	Yes	Yes	Yes
Flexible Components	No	Yes	Yes	Yes
Hot-working Conditions	Yes	Maybe	No	No
Customised Fingers Shape	Yes	Yes	Maybe	Maybe

Table 10: List of requirements per every use case

Analysis of the pilot requirements revealed that all cases require inspection capabilities, component removal and placement functions, and screw handling mechanisms. Consequently, the common

robotic solution presented in this section had to incorporate all these functionalities into a unified design. Specifically, the majority of manipulation tasks may be addressed by adopting suction cup grippers, parallel finger grippers, screwdriver attachments, and holder mechanisms for cleaning hoses or similar tools across all pilot applications.

The initial developed solution consisted of a multi-tool end effector concept featuring different capabilities distributed at 120-degree intervals, including retractable screwdrivers, parallel finger grippers, and suction cups (Figure 33 a). However, this initial solution presented significant volume limitations, making it challenging to operate in tight and compact environments while sacrificing dexterity due to its bulky configuration. After evaluating these limitations, a second solution was developed: a 2-DoF gripper system with active interchangeable fingers that can accommodate various tool attachments, for instance screwdrivers and cutters (Figure 33 b). This approach addresses the spatial constraints and offers excellent modularity. This system, controlled by a locker finger motor, enables each finger to independently adjust its gripping configuration beyond the main gripper motion, offering enhanced versatility for handling different component shapes and sizes while providing precise gripping force management. A close look at this mechanism is depicted in Figure 34.

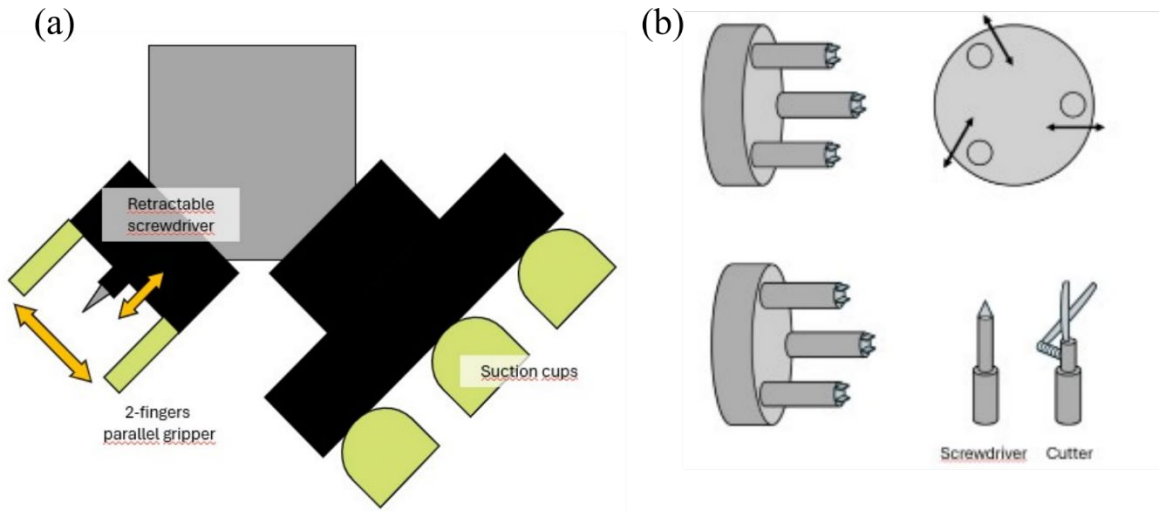

Figure 33: Schematic representation of initial draft of the general RENÉE solution for intelligent end-effectors

The active tip mechanism provides the second degree of freedom through a locker finger system that operates across three distinct positions, transitioning from an open position for maximum opening, through a rest position as the neutral state, to a close position for secure gripping.

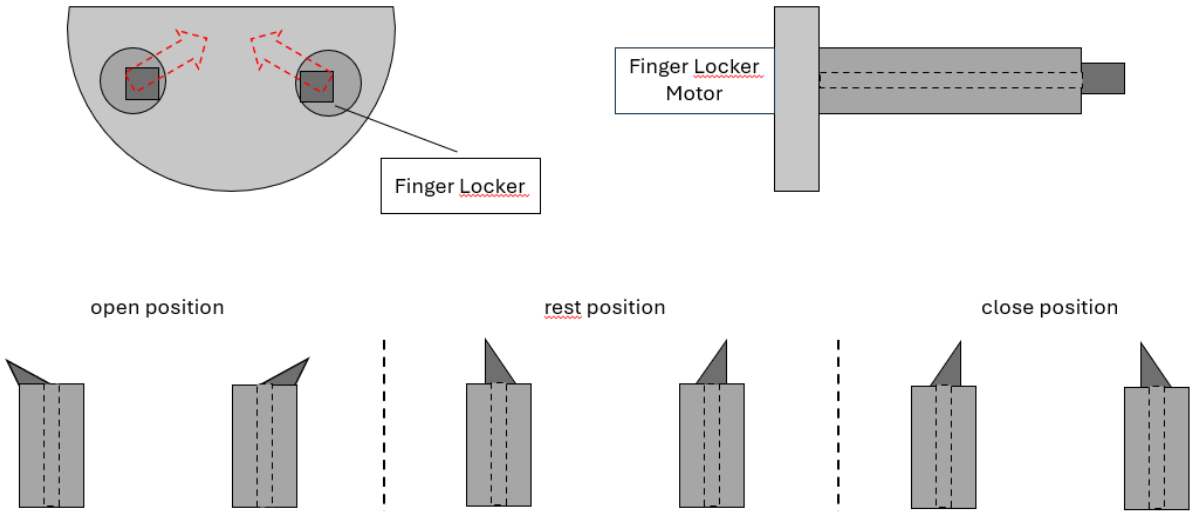

Figure 34: Schematic representation of active tips mechanism designed on the fingers of the end effector, which enables high precision manipulation tasks

The second version of the general solution was further developed to better account for the requirements and needs of the industrial pilot cases. Specifically, the end effector at this stage was modified to incorporate suction cups for handling smooth surfaces, one finger was replaced with an active tool attachment for specialized tasks, and an eye-in-hand camera system was positioned in the palm for enhanced visual feedback. The design remains open to the possibility of installing various sensors such as proximity, slip detection, torque and force sensors, temperature, and pressure sensors as needed, allowing the end effector to be tailored to the specific requirements of each pilot case while maintaining its versatility across all remanufacturing scenarios (Figure 35).

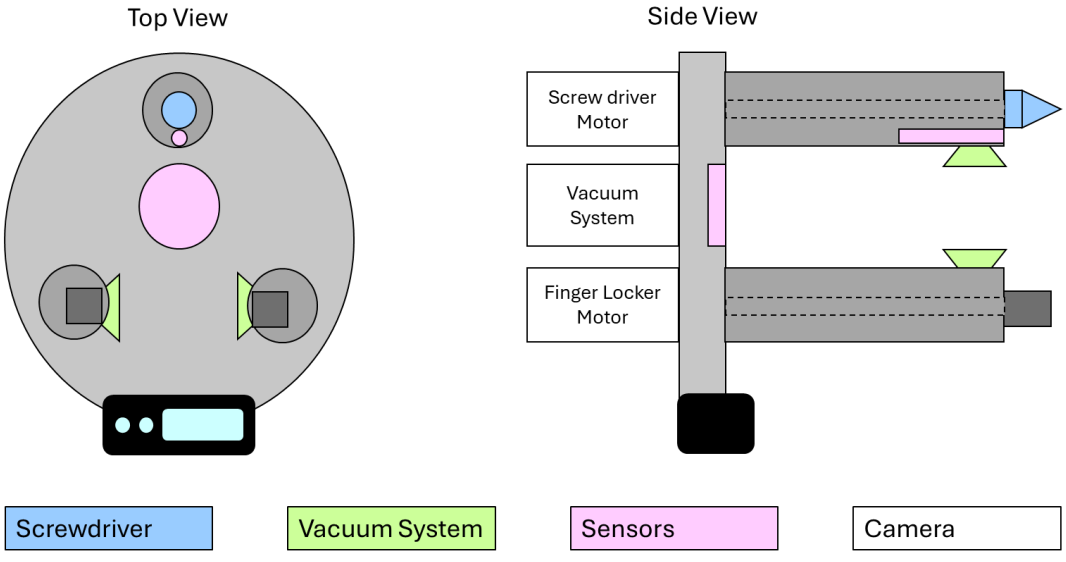

Figure 35: Schematic representation of the updated design of the general solution for the intelligent end-effector

After designing this stage of the end-effector, the list of requirements of the different pilot cases was refined, in order to align with the latest progress and developments of the project. The results are reported in Table 11.

Requirements	EMOTORS	ARCELIK	CAMPETELLA	DECATHLON
Screwdriver	Yes	Yes	No	Yes
Camera	Yes	Yes	Yes	Yes
Suction Cups	Yes	Yes	No	No
Other tools	-	Cutter	Thermal Camera	Steam Cleaner

Table 11: Updated list of features per every use case

Given the complexity of remanufacturing tasks and the diverse automation requirements across different pilot scenarios, the final design concept addresses these challenges through the implementation of an automated finger changing tool mechanism that enables automatic reconfiguration of the gripper capabilities during operation. A schematic representation of the final design of the general RENÉE solution for the intelligent end-effector is depicted in Figure 36. The component in light grey in the figure can be easily both manually or automatically detached from the robot and replaced with the appropriate end-effector.

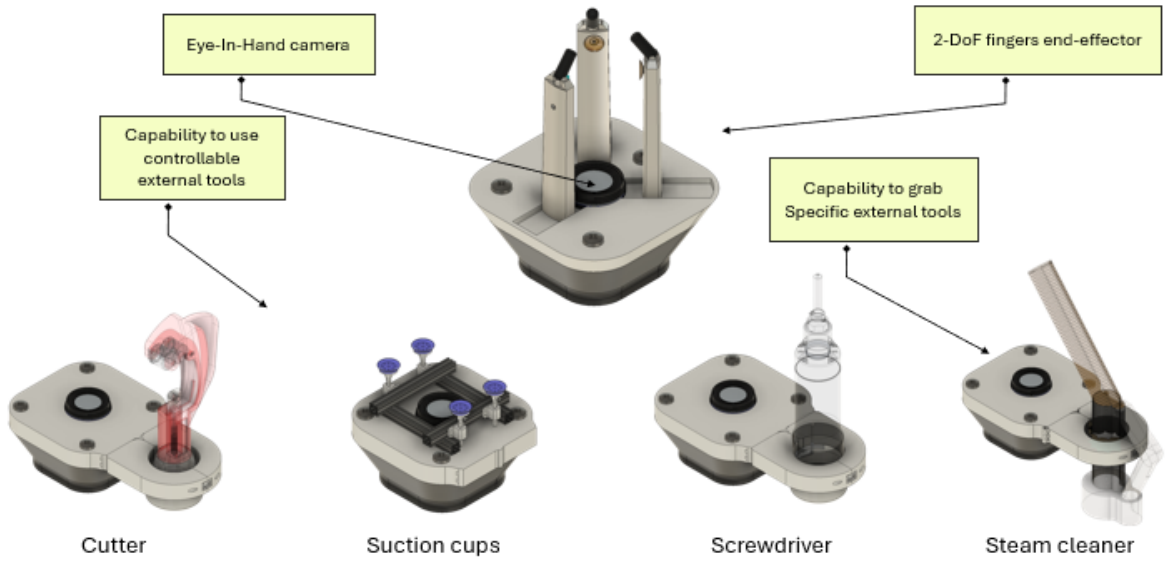

Figure 36: Schematic representation of the final design of the general RENÉE solution for intelligent end-effectors

This innovative approach demonstrates significant potential by integrating multiple specialized tools including cutters, suction cups, screwdrivers, and steam cleaners into a single adaptable platform with an eye-in-hand camera system and 2-DoF finger end-effector. However, the design remains flexible to accommodate pilot-specific needs, with the technological solution open to being deployed as separate, independent grasping systems rather than exclusively through the finger changing tool mechanism. This allows for simultaneous execution of multiple tasks using different specialized end-effectors operating independently, ensuring optimal performance tailored to each remanufacturing scenario without compromising operational efficiency.

The different designs tailored to the specific use cases and having their roots in the general RENÉE solution are described in detail in the following sub-sections.

6.1.1 Electric motor pilot case

The Electric motor use case has been analysed in order to identify the specific features of the disassembly tasks and the managed parts. One of the main requirements is the necessity to manipulate the whole electric motor, moving it to different poses in order to facilitate the screw and connector removal of the different components. Due to the heavy weight of the part, around 50kg, and the few grasping holes and surfaces, it is necessary to adapt the motor grasping gripper to the geometry of the part.

Additionally, the main disassembly task is the screw removal. Nevertheless, it is necessary to integrate a vision system to enable a vision-based fine positioning to overcome misalignments and adaptation to different motor references. The next subsections provide further information about both grippers.

6.1.1.1 Whole Motor Gripper

In order to move the whole motor around the cell, the design of the whole motor gripper makes use of the only three holes available on the motor housing to grip it tightly and support all its weight during the manipulation of the part. The gripper includes two Festo DFM-63 25-P-A-GF pneumatic cylinders with a force up to 1870N. This design enables the safe manipulation of the whole motor, allowing the access to the different screws and connectors to be removed.

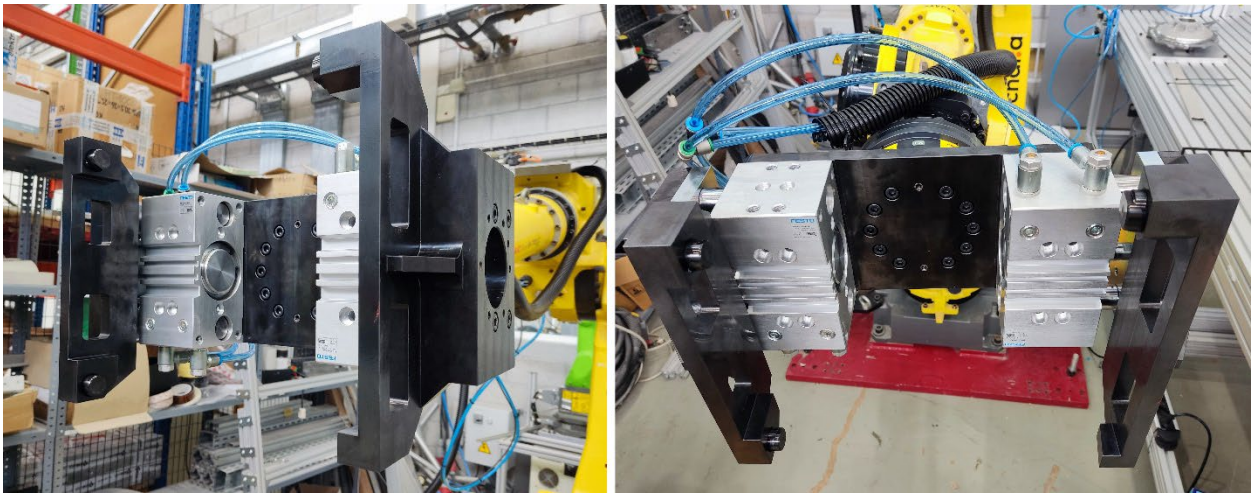

Figure 37: Gripper for complete electric motor manipulation

6.1.1.2 Unscrewing Tool

The unscrewing tool includes an OnRobot Screwdriver, a configurable electric screwdriver able to fasten and unscrew bolts with an additional bolt-finding mechanism able to absorb small misalignments (of around 1-2mm). Additionally, the design is completed with a small Realsense RGBD camera for the implementation of Visual Servoing skills.

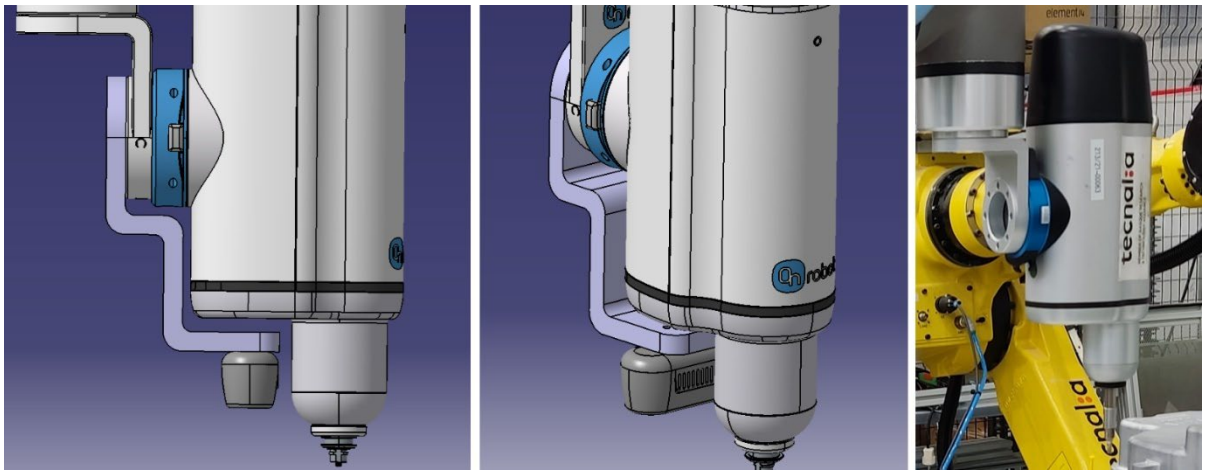

Figure 38: Unscrewing gripper with electric screwdriver and RGBD vision system

6.1.2 Household Appliances pilot case

In the context of the ARCELIK use case, the requirements for the design of the end effectors were initially derived from the analysis of the components that need to be manipulated. The wide variety of the characteristics of these components (weight, size, geometry) led to the design of multiple (four) end effectors combined with a tool changing system enabling the switch between them during execution.

These four end effectors are the following and the requirements/technical specifications of each one are presented in Table 12:

1. **End Effector A:** A multi-tool end effector equipped with fingers, suction cups and a cutter to handle, manipulate and cut small components, like plastic covers, hinges and wires
2. **End Effector B:** A screwdriver to handle unscrewing operations
3. **End Effector C:** An end effector equipped with suction caps to manipulate the doors of the refrigerator
4. **End Effector D:** A heavy-duty end effector equipped with parallel fingers combined with suction caps for the manipulation of the whole refrigerator

Name	Handled Parts	Functions	Capacity
End Effector A	• Covers • Hinges • PCBs • Wires	• Suction • External Holding • Cutting	• Suction Capacity 3kg • Holding Capacity 1kg

Name	Handled Parts	Functions	Capacity
			• External Grasping width 200mm
End Effector B	• Metric & Din Screws	• Screw Tightening • Screw Losing	• Metric and Din Screws up to 35mm length
End Effector C	• Doors	• Suction	• Suction Capacity 30Kg
End Effector D	• Refrigerator	• Suction • External Holding	• Holding Capacity 150kg

Table 12: ARCELIK End Effectors requirements and specifications

6.1.2.1 End Effector A

This end effector was designed to take over the light tasks of the refrigerator disassembly, handling the removal of the small and light parts and the cutting of the wires. The design process followed the generic guidelines provided by IIT with modifications and adjustments tailored to the dedicated requirements of the use case.

The initial design of this end effector was designed and manufactured by TF-CC and initial testing was performed in the LMS testbed (Figure 39). Various issues were identified during the testing related to the mounted on the robot sensor visibility and the volume of the tool creating limitations to robot movements.

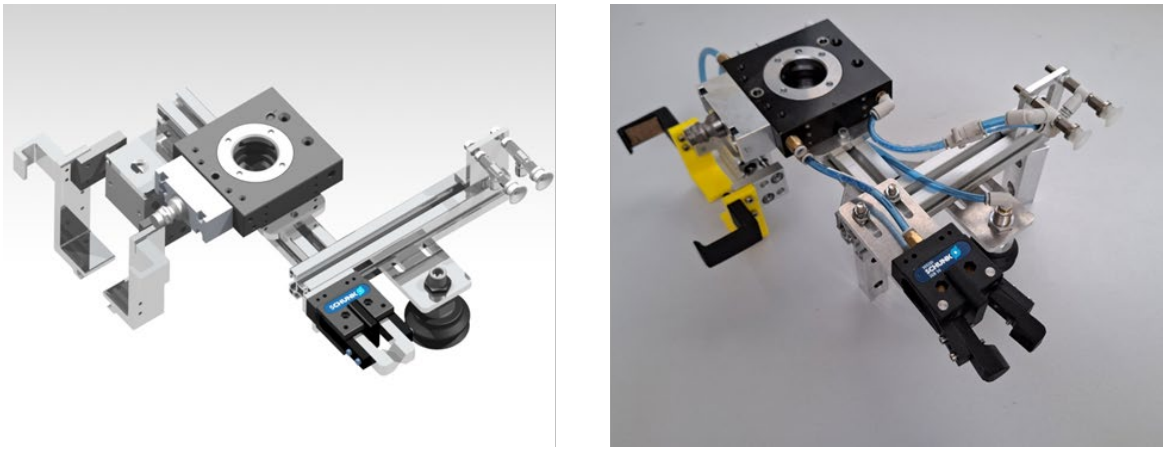

Figure 39: End Effector A 1st Prototype Design & Implementation

Thus, a second version of the end effector was designed based on the generic RENÉE guidelines to mitigate these issues. The updated design (Figure 40) has been tested in a virtual environment to validate its usability and identify any potential accessibility issues that could create.

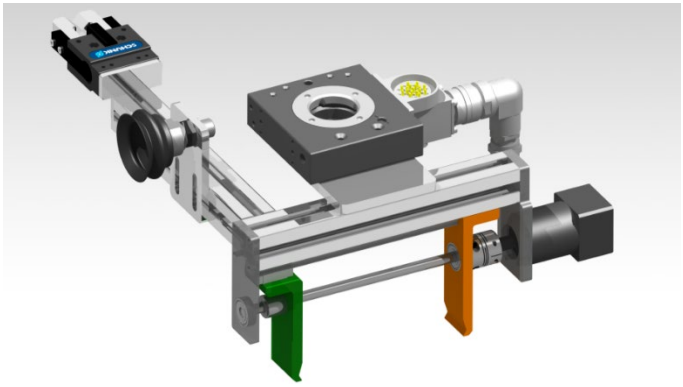

Figure 40: End Effector A 2nd Prototype Design

6.1.2.2 End Effector B

For the unscrewing operations, an automatic screwdriver was selected, due to its integration compatibility with the robot used in the demonstration and the built-in control functionalities it provides (Figure 41).

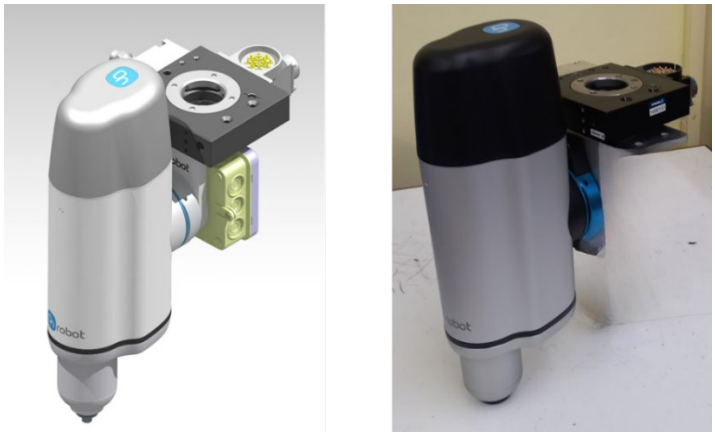

Figure 41: End Effector B

6.1.2.3 End-Effector C

The End Effector C was designed to handle the doors extracted from the refrigerator. The size and the weight of these components require the design of an additional tool, since the multi-tool End Effector A would not be capable to handle them. Thus, the solution presented in Figure 42 was selected. The designed End Effector is equipped with multiple suction caps selected based on their specifications to handle the weight of the refrigerator doors.

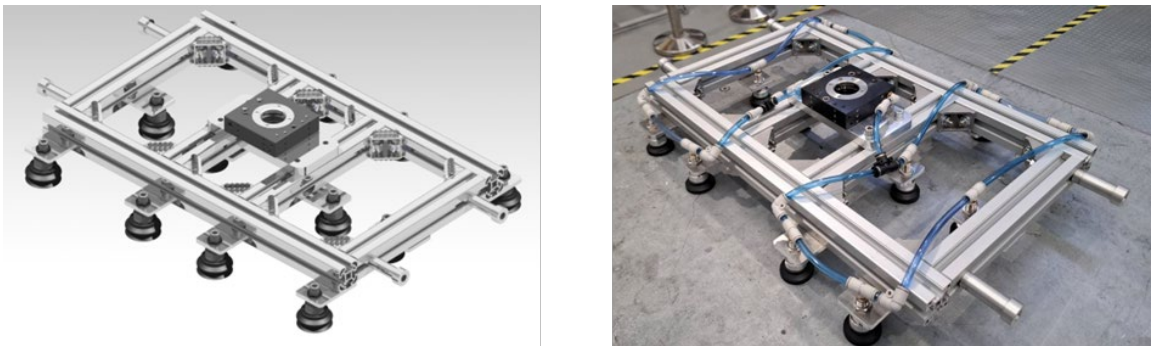

Figure 42: End Effector C

6.1.2.4 End-Effector D

During the second phase of the pre-industrial demonstration that will be built at the TF-CC facilities, the manipulation of the whole refrigerator is envisioned. For this purpose, a heavy-duty end effector has been designed combining a parallel-finger mechanism and suction caps to increase safety and gripping robustness during the manipulation and movement of the refrigerator (Figure 43). This tool will be manufactured during the second phase of the project and will be used by the heavy payload robot envisioned to be integrated into the Arcelik testbed to be installed at TF-CC.

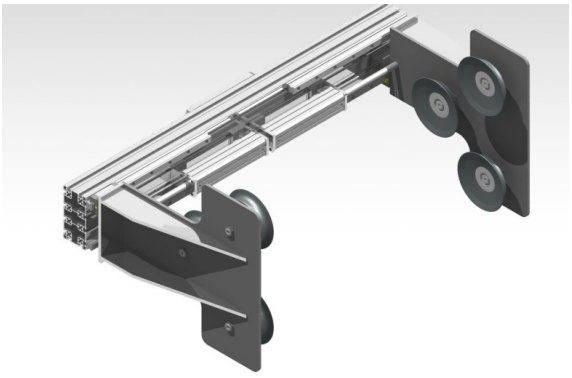

Figure 43: End Effector D

6.1.3 Robots pilot case

The implementation of the sensorised finger changing tool for Campetella's cartesian robot remanufacturing application represents a paradigm shift in automated end effector design, addressing the inherent complexity and variability characteristic of circular economy operations. As illustrated in the schematic representation in Figure 44, the smart end effector system comprises a finger changing tool installed on a collaborative robotic arm together with a 3D camera, demonstrating the comprehensive integration of multiple manipulation modalities within a single platform. The system exhibits autonomous tool switching capabilities amongst four distinct configurations: a two-finger parallel gripper with two degrees of freedom, a custom-shaped tool optimised for handling external tools such as steam cleaner hoses, an array of suction cups arranged in strategic cross-pattern formations, and an electric screwdriver for automated fastening operations.

Figure 44: Schematic representation of the smart end-effector developed for the Campetella use case

The system is comprised of a finger changing tool installed on a robotic arm together with a 3D camera. The system is capable of autonomously switching tool, among a 2-finger parallel gripper with 2 DoF, a custom-shaped tool to best fit the handling of external tool, such as the hose of the steam cleaner, an array of suction cups and an electric screwdriver.

This adaptive architecture incorporates an eye-on-hand vision system positioned on the wrist of the robotic arm, providing real-time visual feedback for precision manipulation tasks, whilst maintaining compatibility with force/torque sensors, and proximity detection. The integrated 3D camera system provides advanced object recognition capabilities, enabling autonomous detection and localisation of screws and diverse components within the remanufacturing workspace. Through real-time image processing algorithms, the vision system accurately identifies target objects and calculates optimal grasping strategies (Section 5.2.3), automatically selecting the most appropriate end effector configuration from the available tool set. The camera-guided approach ensures precise positioning and orientation control during manipulation tasks, whilst simultaneously providing quality assessment feedback to validate successful component handling and placement operations. This technological advancement provides the versatility and reliability essential for industrial-scale circular economy implementation.

The modular design architecture enables seamless integration across Campetella's entire robotic product portfolio, ensuring that the smart end effector solution can be deployed universally throughout their industrial robot production line. This scalable approach allows the finger changing tool system to be readily adapted to different robot models and configurations within Campetella's manufacturing range, from compact precision units to heavy-duty industrial platforms, without requiring model-specific modifications

6.1.4 Bicycles pilot case

The pilot case related to Decathlon has been analysed with the objective of defining an automated procedure for inspecting used bicycles returned to stores and of developing robotic end-effectors capable of performing such operations efficiently and repeatably.

As part of this task, two tools will be developed:

- a tool for thermographic inspection, designed for robotic integration and aimed at inspecting carbon fibre frames;
- Tool for the automatic brake performance check, to be integrated with the test bench developed by STAM.

6.1.4.1 Thermographic Inspection Tool

The thermographic tool is designed to mount both an infrared camera and a heating source directly on the robot flange, as represented in Figure 45.

Figure 45: The end-effector to hold the thermal camera and the heat source

The integration of the heating element directly on the tool ensures uniform and symmetric heating of the component, which is essential for obtaining reliable and consistent thermal images. The structure of the tool will be designed to avoid interfering with the robot's motion, ensuring the flexibility needed to adapt to frames of different geometries.

6.1.4.2 Brake Control Tool

The brake inspection tool has been conceived to reproduce, in a controlled and repeatable manner, the standard tests used in production—now applied also to used bicycles. The aim is to assess the braking system's performance under simulated dynamic conditions.

Proposed procedure:

1. The bicycle is placed manually by the operator on a motorised roller.
2. The bicycle is securely fixed in position using dedicated clamping systems.
3. The roller is activated, imparting a controlled rotational speed to the rear wheel to simulate riding conditions.

Once the wheel reaches the desired speed, the operator uses a manual gripper to activate the brake lever. This specially designed gripper applies controlled, incremental force to the brake lever, while a torque sensor mounted on the roller measures the resulting braking force. The objective is to assess the braking performance of the system by comparing the measured values to safety reference thresholds.

The force values and performance limits will be inspired by relevant standards (e.g., ISO 4210-4), but will serve only as guidelines for tool calibration, not as strict specifications.

The gripper will be developed to combine simple construction, safe operation, and precise force control. The tool is manually positioned by the operator on the brake lever and allows both secure holding of the handlebar and effective activation of the brake lever. To ensure maximum operator safety, the gripper will feature a safety push-button: the actuator will only operate while the button is actively pressed. If the button is released, the gripper will immediately disengage, eliminating any risk of unintended clamping.

A possible solution, as shown in Figure 46, involves a pneumatic cylinder controlled by a proportional valve, allowing fine regulation of air pressure and, consequently, real-time modulation of the applied force on the brake lever. This is essential for applying incremental loads as required by the test procedure.

The system also includes a linear position transducer coupled to the cylinder piston, which measures the brake lever's displacement during activation. This provides information on lever travel and actuation speed, which can help detect mechanical anomalies such as excessive play or irregular response.

Figure 46: A potential solution to perform inspection on the brake handles

This configuration makes it possible to faithfully and repeatably replicate real test conditions, making the system suitable not only for production environments, but also for after-sales and retail contexts, where fast, automated and objective checks are required. The use of standard components such as pneumatic actuators, proportional valves, and linear sensors also ensures easy maintenance and scalability for future upgrades.

7 Conclusion and Future work

This deliverable presented the initial prototypes development of the Robot Skills & Flexible Production Modules for Remanufacturing within the RENÉE framework. The work accomplished includes the design and implementation of a modular Robot Skills Library that encapsulates remanufacturing operations as reusable, parameterizable skills, the development of flexible production modules with resource control frameworks and hardware abstraction layers, the creation

of AI-enhanced perception systems capable of handling core variability through advanced computer vision and machine learning techniques, and the design of intelligent end effectors with adaptive gripping capabilities for diverse component manipulation across four industrial use cases (EMOTORS, ARCELIK, CAMPETELLA, and DECATHLON).

The key advancement achieved beyond the state of the art is the successful integration of modular robot skills with AI-driven perception systems that can autonomously adapt to the inherent variability in remanufacturing processes, significantly reducing programming complexity while enabling scalable automation for high-mix, low-volume remanufacturing scenarios.

The next steps involve refining these initial prototypes based on integration testing feedback, developing advanced learning capabilities for skill adaptation, enhancing the perception modules with more sophisticated defect detection algorithms, and conducting comprehensive validation testing in the integrated testbeds before final deployment to industrial pilot sites. These developments will culminate in the final prototype deliverable D5.2, which will demonstrate the full operational capabilities of the RENÉE remanufacturing system across all four industrial use cases.

References

BehaviorTree.CPP. (n.d.). Retrieved June 20, 2025, from https://www.behaviortree.dev/

Biancucci, Massimiliano. (2025). *COIGAN: Controllable Object Inpainting through Generative Adversarial Network for Defect Synthesis in Data Augmentation* [Python]. VRAI. https://github.com/vrai-group/COIGAN-controllable-object-inpainting (Original work published 2024)

Isaac Sim. (n.d.). NVIDIA Developer. Retrieved June 20, 2025, from https://developer.nvidia.com/isaac/sim

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection* (arXiv:1506.02640). arXiv. https://doi.org/10.48550/arXiv.1506.02640

Ros-controls/ros2_control. (2025). [C++]. ros-controls. https://github.com/ros-controls/ros2_control (Original work published 2017)