A Reconfigurable ROS2-Based Architecture for Robotic Management of Apparel and Accessories

A Reconfigurable ROS2-Based Architecture for Robotic Management of Apparel and Accessories

Andrea Bonci
Dept. of Information Engineering
Polytechnic University of Marche
Ancona, Italy
a.bonci@univpm.it

Alessandro Di Biase
Dept. of Electrical and Information Engineering
Polytechnic of Bari
Bari, Italy
a.dibiase8@phd.poliba.it

Marina Indri
Dept. of Electronics and Telecommunications
Politecnico di Torino
Torino, Italy
marina.indri@polito.it

Matilde Zampolini
Dept. of Electronics and Telecommunications
Politecnico di Torino
Torino, Italy
matilde.zampolini@polito.it

***Abstract*—Robotic processing of apparel items and accessories is still challenging because such products exhibit strong variability in geometry, material composition, deformability, and structure. Planar garments, multilayer textiles, footwear, and other apparel-related items require different sensing, interpretation, and execution strategies, limiting the generalization of highly task-specific solutions. This paper presents a reconfigurable ROS2-based robotic architecture for heterogeneous apparel items and accessories. The framework integrates visual perception, semantic feature extraction, coordinate transformation, task planning, motion planning, execution, and feedback handling into a unified pipeline deployable on a single-arm robotic platform. The aim is to provide a flexible, multi-task architecture for heterogeneous items and operations in realistic industrial workflows. Rather than targeting a single product category or operation, the architecture analyses the visual and geometric characteristics of the target item and configures the processing workflow accordingly. The framework is here instantiated for selective disassembly, a demanding circular economy use case requiring accurate localization and treatment of seams and heterogeneous components. A case study on jumpers and footwear demonstrates how the same architectural backbone can support substantially different item geometries and processing requirements.**

***Index Terms*—Task-Adaptive Robotics, Visual Perception, Collaborative Robotics, ROS 2, Circular economy.**

I. Introduction

The automation of operations involving apparel items and accessories is becoming increasingly relevant in both industrial and circular-economy contexts [1]–[3]. These products exhibit wide variability in geometry, material composition, stiffness, layering, and assembly details, making it difficult to design robotic systems that generalize across multiple product categories and tasks. A practical solution should therefore not be conceived as a task-specific pipeline, but as a reconfigurable architecture capable of perceiving heterogeneous items, interpreting relevant structures, and adapting the execution strategy accordingly.

This need is particularly evident when target objects range from nearly planar garments to fully three-dimensional items, such as shoes and structured accessories. Inspection, seam following, trimming, component removal, selective disassembly, and sorting share common requirements, including visual and geometric perception, identification of task-relevant features, image-to-robot coordinate transformation, task sequencing, motion generation, and online adaptation. However, these capabilities are often addressed separately or for narrowly defined use cases, limiting the transferability of existing solutions.

Although apparel manufacturing is progressively adopting automation, current solutions often rely on dedicated machines for individual processes [4], [5]. Sewing machines provide a relatively high degree of flexibility, but remain restricted to specific operations and quasi-planar tasks [6]–[8]. Consequently, complete production workflows often require multiple specialized robotic cells, increasing system complexity, cost, and reducing adaptability to changes in product types or processing requirements.

Robotic manipulation of apparel-related objects has been widely investigated, particularly for garment perception [9]–[11], grasping and unfolding [12], co-manipulation [13]–[16], and for the detection of components such as seams, stitches, and fasteners [17], [18]. While these contributions are valuable, integrated architectures capable of handling heterogeneous items across different operational requirements remain limited [19]. The key challenge is therefore the design of a reusable perception-to-action framework, that can be reconfigured according to item geometry, structure, and processing objective.

In this paper, we present a reconfigurable ROS2-based

Project co-funded by: the European Union – Next Generation Eu - under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 1 Investment 4.1 - Decree No. 118 (2nd March 2023) of Italian Ministry of University and Research - Concession Decree No. 2333 (22nd December 2023) of the Italian Ministry of University and Research, Project code D93C23000450005, within the Italian National Program PhD Programme in Autonomous Systems (DAuSy).

robotic architecture [20] for heterogeneous apparel items and accessories. The key innovative solution provide: 1) a unified framework integrating perception, semantic feature extraction, coordinate transformation, task planning, motion planning, execution, and feedback; 2) an adaptive processing strategy that analyses the observed product and activates the most appropriate processing workflow; 3) the integration of task sequencing and motion generation.

The architecture is validated on selective disassembly of jumpers and shoes, a demanding benchmark because requiring accurate localisation of seams, heterogeneous fasteners, and both planar and three-dimensional structures. The framework is implemented on a single-arm collaborative robotic setup, making it compatible with small- and medium-scale facilities, supporting the European strategy for a sustainable and circular textile sector [21], [22].

The remainder of this paper is organized as follows: Section II describes the proposed architecture; Section III presents the experimental case study; Section IV draws conclusions and directions for future works.

II. Architecture

The proposed ROS2 architecture covers the full pipeline from apparel-item detection to task execution, addressing modularity, scalability, and real-time performance. The architecture, shown in Figure 1, is composed of eight main blocks (in light blue), along with two additional blocks (in light green) dedicated to the management of external devices.

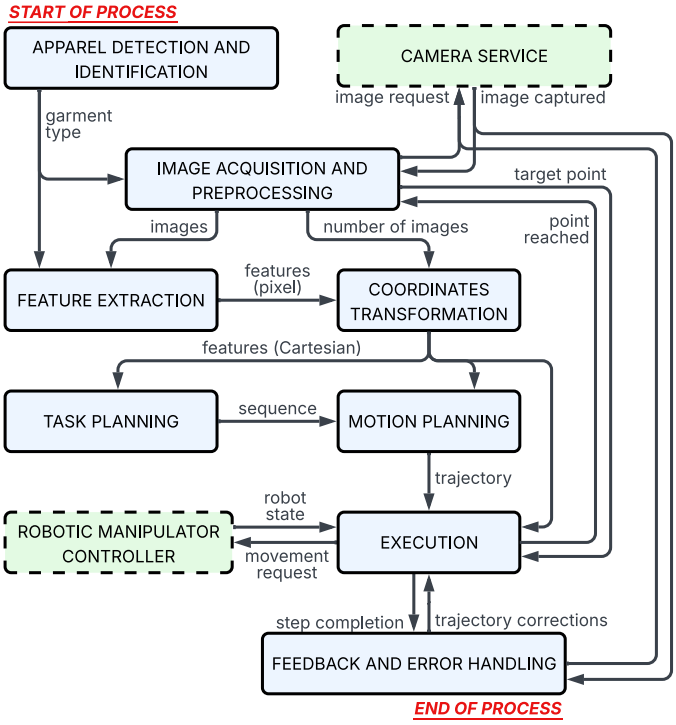


Fig. 1: ROS2 Architecture: blue blocks represent the high-level nodes, green blocks represent the hardware management nodes and the arrows show the publisher-subscriber connections.

Each block is implemented as a ROS2 node and exchanges data through typed topics. The perception pipeline includes four blocks: garment-type recognition, image acquisition, feature extraction (relevant to the process), and pixel-to-robot coordinate transformation. The planning and execution pipeline handles the last four blocks: task sequencing, motion generation, trajectory execution, and error management. Specifically, the best sequence of tasks is derived, the motion of the robotic arm's end-effector is planned and then executed, and finally, the last block is used for error management. The architecture allows the integration of existing robotics frameworks. Standard solutions are retained when adequate, such as MoveIt for execution, whereas custom modules are introduced only if apparel-specific requirements cannot be directly supported by existing frameworks, such as operation-dependent task sequencing. The image acquisition and execution nodes depend on the integrated hardware. The feature extraction node is highly customizable to allow the implementation of the optimal neural network depending on the context. The task planning node implements the core logic of all processes, depending on the given objectives. Although the reconfigurable implementation modules need to be adapted to different apparel items and operations, the architecture is based on fixed communication interfaces.

It is worth noting that only a subset of nodes depends on hardware-specific parameters, namely those responsible for image acquisition and preprocessing, coordinate transformation, and execution. The former depend on camera parameters, while the latter depends on the robotic manipulator. The use of a configuration file, loaded by the launch file, simplifies parameter setting and improves maintainability.

The following subsections describe the characteristics and operation of the various blocks in detail.

A. Perception pipeline

1) Detection and Identification:

Publishing topic: *garment_type*

In the first phase of the process, the first node identifies the garment type and uses this information to select the appropriate feature-extraction, task-planning, and prioritized execution strategy. In the current release, the garment type is predefined by the user, which improves efficiency when different product categories are not mixed in the same process. If mixed items are processed, this node can be replaced by an automatic classification module, for instance based on neural networks [23], [24]. The selected garment type is published as a String message on the *garment_type* topic.

2) Image acquisition:

Subscribed topic: *garment_type, point_reached*

Publishing topic: *images, number_of_images, target_point*

Service: *camera_service*

The node handling image acquisition, the second block in Fig 1, subscribes to the *garment_type* topic, and consequently triggers the corresponding operations required for image acquisition implemented via OpenCV and device-specific drivers. The goal of this phase is to provide to the

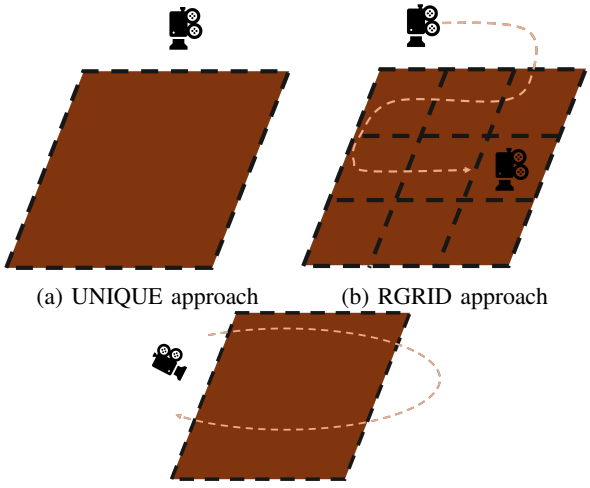
robotic system geometrical and qualitative information on the surrounding environment to be used both for motion planning and control.

The main feature of the vision layer is the camera placement. The vision layer can operate in either eye-to-hand or eye-in-hand configuration. The former provides a fixed view of the workspace but may suffer from occlusions, whereas the latter changes the viewpoint with the manipulator motion (the camera is mounted on the manipulator's wrist) and is therefore better suited to complex geometries. In this work, the eye-in-hand configuration is adopted due to its better accuracy.

Furthermore, two different types of cameras can be used: RGB cameras that recognize specific objects from data collected on the same plane, or depth cameras that allow to get richer information about objects' position and dimensions.

In the developed architecture, different processes for image acquisition are applied, depending on the type of garment:

- UNIQUE approach: if the object can be considered as two-dimensional, a single image of the entire object lying on a flat surface is taken from above (Fig. 2a);
- RGRID approach: if the object has details that may not be clearly identified from a single image of the entire garment, e.g., not easily visible stitching, several close-up images of the garment are taken, and the overall image is then reconstructed as a grid (Fig. 2b);
- 360D approach: if the object is intrinsically three-dimensional, the camera must rotate around the object to reconstruct its specific shape (Fig. 2c), e.g., to detect the lateral curvature of a shoe.



(a) UNIQUE approach (b) RGRID approach

(c) 360D approach

Fig. 2: Approaches for acquiring images: brown denotes the workspace, orange the movement of the camera.

Regardless of the method used to capture images of the garment, proper coordination of the robot's motion and the capture of images from appropriate viewpoints is always crucial in this activity. The wrist-mounted camera is moved to ensure complete workspace coverage, prioritizing overlap over uncovered regions. Two functions are alternatively executed: the first one moves the robotic arm via the *target_point* topic, while the second one handles image acquisition, requesting the camera service. Each function is triggered only upon completion of the other, ensuring that motion and image acquisition are mutually exclusive and never overlap. The pose of the robot reached at each step is received by the node through the *point_reached* topic.

Despite the specific implementation, the message types of the topics, where sensor data are published, should be as standard as possible, in order to exploit existing ROS2 functionalities, like the ones included in the *sensor_msgs* ROS2 package, dealing with sensor devices. For example, for an RGB camera, there is the *sensor_msgs/msg/Image* message type, which contains an uncompressed image with a header with fields such as timestamp, image dimensions in pixels and encoding (e.g., rgb8), and a body that is the actual data matrix.

In the developed architecture, image acquisition is implemented as a ROS2 service that is activated at the beginning of the process. Specifically, the service receives the requested resolution and returns the image as a *sensor_msgs/msg/Image* message and a boolean parameter that identifies the success of the operation.

At the end, the node publishes the acquired images to the *images* topic and the information on the number of required images to the *number_of_images* topic. A new message type was created to pass in a single message the image, the resolution, the pose of the robot arm from which the image was taken, together with a string identifying the detected features, the coordinates in pixels and those converted into the real world frame that will be handled by the subsequent nodes.

3) Feature extraction:

Subscribed topic: *garment_type, images*

Publishing topic: *features_pixel*

The third node handles the main activity of the process. Once the images have been acquired, they must be analysed by a model that extracts the image regions where tasks have to be executed. The goal of this phase is therefore to automate the recognition process of such parts in order to define the entire specific process. The architecture imposes to publish geometrically localized features via the standardized *features_pixel* topic. Consequently, various neural networks can be integrated. Operative tasks may depend on colour, geometry, texture, or material composition. Since these properties cannot generally be separated by simple thresholding, especially in disassembly-oriented applications, neural-network-based perception is adopted to extract task-relevant regions from the acquired images. Deep learning approaches have been widely investigated in recent decades, given their capability to lead to end-to-end learning where raw input data can be mapped to the desired output [25]. This contrasts with classic machine learning algorithms, where models cannot automatically learn complex and abstract feature representations from raw data, requiring manual feature engineering.

The identification of task-relevant regions is formulated as a multi-class semantic segmentation problem [26]–[28], since pixel-level masks provide more accurate geometric informa-

tion than bounding boxes for planning cutting or manipulation trajectories [29].

Based on the information on the garment type received from the *garment_type* topic, a dedicated neural network is selected to process the acquired images. The extracted features, together with their pixel coordinates, are merged with the data from the *images* topic and published on the *features_pixel* topic.

To have a unique architecture for all types of garment, it is necessary to import all the libraries required for the various neural networks to be used. First, all neural networks must be brought under the same Pytorch or Tensorflow library, which are the two main open-source frameworks for deep learning. If one neural network was trained with Pytorch and another with Tensorflow, the weights of one of the two must be converted to avoid compatibility issues between the required versions of torch, tensorflow, openCV, and numpy. It is also necessary to pay attention to the Python version required by the neural network models, as this could create compatibility issues with the ROS2 version. If the models require Python 3.10, ROS2 Humble is sufficient; otherwise, ROS2 Jazzy is required to use Python 3.12.

4) Coordinate transformation:

Subscribed topic: *features_pixel, number_of_images*
Publishing topic: *features_cartesian*

The camera calibration procedure maps the pixel coordinates obtained from the perception node into spatial coordinates in the robot reference frame. In addition to the TF2 library, the transformation relies on two calibration terms:

- Camera Intrinsics: the matrix $\mathbf{K} \in \mathbb{R}^{3,3}$, containing focal length, principal point, and screw factor, maps the 2D image pixels to the camera coordinate frame.
- Camera Extrinsics: the homogeneous transformation matrix $\mathbf{T}_{ee}^{cam} \in \mathbb{R}^{4,4}$ maps the points from the camera frame to the end-effector frame.

Since manipulator motions are defined with respect to the robot base frame, each detected pixel coordinate (u, v) is converted into the corresponding 3D spatial coordinate (x, y, z), as described in [30].

As introduced in Section II-A2, some objects may require multiple close-up images with higher resolution to identify specific parts. In this case, the node also merges the spatial coordinates obtained from the different views to reconstruct the complete feature profile. Therefore, it subscribes to the the *number_of_images* topic and starts the fusion only after all required images have been received. Partial overlaps are handled by removing duplicate coordinates according to a proximity threshold, while preserving segment direction and point ordering.

Upon receiving each message from the *features_pixel* topic, the robot joint configuration is used to compute the current forward-kinematics transformation, while the intrinsic matrix $\mathbf{K}$ is evaluated from the camera resolution and calibration parameters. The resulting coordinates are fused and published as a single message on the *features_cartesian* topic.

B. Planning and execution pipeline

1) Task Planning:

Subscribed topic: *features_cartesian*
Publishing topic: *sequence*

The task-planning node converts the detected features into actions and generates an ordered execution plan. This node is highly dependent on the application and garment context. The strategy is independent of the specific apparel type and combines semantic priorities, such as the risk of damaging the item or the logical order of assembly/disassembly operations, with performance-oriented criteria, including path length, execution time, energy use, and tool-change minimization. Beyond semantic priority, the robotic manipulator minimizes a cost functional to optimize the overall performance. This choice is driven by two competing objectives. First, reducing the total path length lowers energy consumption and execution time, thereby improving the economic efficiency. Second, and more subtly, prioritizing smaller impact tasks early in the sequence helps to preserve the structural integrity of the workpiece. Indeed, if the system produces fragments with a small surface area, these would be more difficult to stabilise with supports, increasing the risk of poor quality in subsequent operations. In this perspective, task planning is a mechanism for improving the effectiveness of the overall robotic process [31].

The corresponding ROS2 node subscribes to the *features_cartesian* topic to receive the coordinates of the regions where the identified actions must be executed. Each message contains a list of Cartesian points arrays, each associated with a detected feature. These features are mapped to one or more corresponding actions, allowing multiple operations to be assigned to the same region when required. Upon reception, the node formulates and solves an optimization problem to determine the execution order. Problems of this type can be interpreted within the framework of integer and combinatorial optimization, such as the Traveling Salesman Problem (TSP), which is typically addressed through exact methods including branch-and-bound [32], cutting planes [33], and branch-and-cut approaches [34]. When priority constraints are involved, the problem can be extended to precedence-constrained formulations, such as the Sequential Ordering Problem [35], or modeled through weighted objective functions [36]. More generally, it can be framed as a multi-objective optimization problem, where competing criteria are handled in a hierarchical or weighted manner [37]. The resulting ordered sequence of features, and extra instructions (e.g. change tool, place or remove item), is then published on the *sequence* topic.

2) Motion Planning:

Subscribed topic: *features_cartesian, sequence*
Publishing topic: *trajectory*

This node generates the Cartesian trajectory required to execute the ordered actions on the apparel item. The trajectory alternates operational segments, which follow the prescribed task paths, and transition segments, which connect consecutive operations at a fixed stand-off distance h from the workpiece surface to avoid unintended contact with adjacent regions.

During transition segments, the end-effector orientation is not strictly constrained, whereas along operational segments it becomes task-critical, for example during cutting, sewing, or stretching operations. Since the operational orientation is derived from garment features detected by the perception pipeline, it may be affected by camera noise, neural-network prediction errors, and pixel-to-Cartesian re-projection artifacts. To avoid oscillatory tool motions and preserve task quality, the orientation signal is low-pass filtered before trajectory construction. Trajectory generation is triggered once both the *feature_cartesian* and *sequence* topics messages are available. The node associates each ordered action with the corresponding Cartesian feature, generates the required operational and transition segments, and includes predefined task parameters, such as loading/unloading poses and tool-change locations. The resulting trajectory is then published on the *trajectory* topic.

3) Execution:

Subscribed topic: *target_point, features_cartesian, trajectory, trajectory_corrections, robot_state*

Publishing topic: *movement_request, point_reached, step_completion*

In the current implementation collision-free motion planning and inverse kinematics are delegated to the MoveIt2 framework. The execution node interfaces the planned trajectory with the robot controller and supervises its completion. Depending on the controller capabilities and the desired control level, commands can be issued as Cartesian trajectories, joint trajectories, joint velocities, or torques. The Cartesian representation is retained to monitor deviations from the nominal operational path and to support closed-loop corrections. In all cases, the node continuously monitors the robot state, expressed as joint configuration or end-effector pose, to detect motion completion and provide runtime information for supervision and fault-detection mechanisms [38].

Equally critical is the mechanical fixturing of the garment during operation. Several support strategies are applicable depending on the item geometry and the operational context:

- *Vacuum table*: a suction surface that holds flat garments in place without mechanical contact, well-suited for two-dimensional items, such as jackets or jumpers. However, the energy consumption of the vacuum system is non-negligible and may increase the overall operational cost
- *Clamping bars*: mechanical bars that press the garment against the work surface, providing robust fixturing without energy consumption.
- *Shaped mannequin or last*: a form that matches the geometry of the item, particularly effective for three-dimensional objects. For footwear, mounting the shoe on a foot-shaped last is the recommended approach, as it provides a stable reference frame and allows rotation of the item without occluding any region to be cut.

As described in Section II-A2, this ROS2 node is responsible for executing the robot motion for image acquisition by subscribing to the *target_point* topic. The node has two execution roles. During image acquisition, it receives target poses from the *target_point* topic, sends motion requests through *movement_request*, and publishes the reached pose on *point_reached* topic once completion is confirmed from *robot_state* topic. During task execution, it receives the planned trajectory from the *trajectory* topic, forwards it to the external controller, and monitors *robot_state* to determine action completion by comparison with the Cartesian features received from *features_cartesian* topic. Each completed action is notified through *step_completion* topic, enabling the feedback node to provide updated trajectories through *trajectory_corrections* topic when local corrections are required. When all actions have been completed, a final empty message is sent on *step_completion* topic.

4) Feedback and error handling:

Subscribed topic: *step_completion*

Publishing topic: *trajectory_corrections*

Service: *camera_service*

The geometry of the garment may change locally during processing, even when fixturing maintains the item in its nominal position. This node therefore verifies the consistency between the current workpiece state and the planned trajectory, updating subsequent tasks if a displacement is detected. Re-executing the full perception and planning pipeline after each operation would introduce significant computational overhead and latency. Instead, the adopted strategy performs a lightweight geometric consistency check at every step, comparing the current state of the workpiece with the previously computed plan. Only the trajectory of the immediately subsequent task is updated, while the global plan is preserved. This approach represents a deliberate trade-off between replanning accuracy and computational efficiency, and aligns with the step-by-step reasoning that characterizes the human-inspired design of the overall architecture.

When triggered by the *step_completion* topic, the execution flow distinguishes three situations: successful completion, recoverable errors, and unrecoverable errors. Successful completion triggers the next planned action, recoverable errors invoke the local trajectory-correction procedure, whereas unrecoverable errors terminate the process and notify the supervisory layer. In case of recoverable errors, the node acquires a local image through *camera_service*, detects the relevant feature, and publishes the updated trajectory on *trajectory_corrections* topic. Since the robot is already positioned near the processed region, image acquisition, feature extraction, and coordinate transformation are integrated locally, making the corrective update faster than full replanning.

For improved adaptability and maintainability, a boolean flag included in the messages exchanged by the perception nodes allows this node to replicate the same coordinate transformation pipeline, while distinguishing between corrective updates and full planning outputs. If the received message contains no features, or if the newly acquired image contains unmanageable errors, the node terminates the process.

III. Case Study:

The proposed architecture is designed to handle a wide range of garment types. The considered case study focuses on jumpers and shoes, which represent examples of two-dimensional garments and three-dimensional objects, respectively. The goal is the separation of the heterogeneous materials they are made of for their future recycle; in the case of shoes, the sole must be separated from the upper, whereas for jumpers, the cuffs, hem, and neckline must be removed, because they contain elastic components. Disassembly was selected as a representative application in the real world, even if only a partial experimental validation is allowed in university laboratories, because cutting tools cannot be mounted on the robot end-effector for safety reasons.

(a) The jumper. (b) The shoe.

Fig. 3: The considered test cases.

The system hardware consists of a Ufactory xarm6 robotic manipulator and an Intel RealSense depth camera. The ROS2 architecture is deployed on a laptop running Ubuntu 24.04 with ROS2 Jazzy.

The implemented neural networks include U-Net, U2-Net, and SAM3 (Segment Anything Model 3) [28], requiring the following software environment configuration:

- Python 3.12 or higher
- NumPy 1.26.4
- PyTorch 2.7 or higher

The carried out tests allow to make a preliminary but significant validation of the entire perception pipeline, and partially of the planning and execution one. The tests are performed in a university lab, avoiding the last execution phase, in which the garments should be actually cut, for safety reasons (a dedicated tool for safe cutting is under construction). In must be noted that the execution of such phase is not mandatory to evaluate the correct working of the proposed architecture, but it would be useful only to assess the specific recycling goal. The implementation and execution of the various nodes/phases are detailed hereafter.

Apparel Identification

As discussed in Section II-A1, the garment type is assumed to have been chosen by the user, selecting the shoe or the jumper option. The node publishes the type selected by the user as a `string` to the *garment_type* topic.

Image Acquisition

The callback associated with the *garment_type* topic generates a list of target poses in a fixed workspace around the item. The image acquisition strategy depends on whether the garment is three-dimensional (shoe) or predominantly planar (jumper). For planar garments, a single top-view image is sufficient, provided that key regions such as cuffs, hem, and neckline are fully visible (Fig. 2a). For shoes, it includes two lateral viewpoints (left and right sides), covering its geometry (Fig. 2c). Since the 360° acquisition requires camera motion, an eye-in-hand configuration is adopted (see Section II-A2). The required number of images is then published as an `Int32` message on the *number_of_images* topic. Subsequently, the first motion is requested by publishing the initial target pose on the *target_point* topic. Once the motion is completed, the callback associated with the *point_reached* topic is triggered. The actual reached pose is stored, and the camera service is invoked to acquire an image. Upon reception, the image is published using a custom message type, `PhotoData`. The `PhotoData` message includes the following fields: a `sensor_msgs/Image` containing the captured image; an array `Line[]` (custom type), where each element consists of three `float64[]` arrays representing the x, y, and z coordinates, for both pixel and Cartesian lines; a `float64[6]` array representing the acquisition pose; and an `int32[2]` array specifying the image resolution (width and height). This process is repeated until the established number of acquired images is achieved.

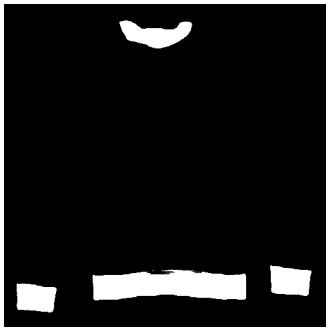
Feature Extraction - Seam and Region Detection

The node adapts its processing strategy according to the garment type received via the *garment_type* topic. For jumpers, a U-Net segmentation network is used to identify the stretch fabric regions to be removed (see Fig. 4a). In parallel, U2-Net is employed to isolate the garment from the background. The boundaries between each target region and the surrounding fabric are then extracted, thinned to a one-pixel skeleton, and converted into continuous cutting lines (see Fig. 4b). The single image received on the *images* topic is processed, and the resulting pixel coordinates are stored in the `Line[]` array before being published on the *features_cartesian* topic.

For footwear, the system processes multiple images using SAM3; the prompt "stitching" has been selected for its high accuracy in detecting seams (see Fig. 4c and 4d). Since individual stitches may appear in more than one image, a reconstruction step consolidates these detections into a consistent representation in the inertial reference frame, according to the coordinate transformation described in the next section. Each image received on the *images* topic is processed independently, its pixel coordinates are stored in the corresponding `Line[]` array, and the resulting messages are published on the *features_cartesian* topic.

Coordinates Transformation

Upon reception of each image from the *features_pixel* topic, the pixel coordinates z_d, u, v stored in the message are converted in Cartesian coordinates, as described in Section II-A4. The information provided by the message is used to com-



(a) Identified regions by UNET

(b) Extrapolated lines of the jumper

(c) Identified regions by SAM3

(d) Extrapolated lines of the shoe

Fig. 4: Steps for determining cutting lines: first identification of the regions, than isolation of the boundaries.

(a) Sequence of the jumper: cutting segments in gray and transition segments in red.

(b) Sequence of the shoe: the unique cutting segment in red.

Fig. 5: The optimal sequence of directed segments.

pute the homogeneous transformation T_{base}^{ee}, and the intrinsic matrix **K**, while the fixed extrinsic transformation T_{ee}^{cam} is constant. In the case of footwear, stitch segments exhibiting spatial continuity across different views are identified and merged into consistent, unified lines in the common reference frame. All the lines in Cartesian coordinates are stored in the array of `Line[]` of a new single `PhotoData` message, and published to the *features_cartesian* topic.

Task Planning

In this case study, no prioritization is imposed among the different cutting operations. As a result, the optimization problem reduces to a TSP, whose objective is to minimize the overall execution time by reducing the total path length.

Upon receiving data from the *features_cartesian* topic, the corresponding callback extracts the first and last points (endpoints) of each cutting line. The problem is then formulated as a complete graph, where each node represents an endpoint. Edges connecting the two endpoints of the same cutting line are assigned zero cost, enforcing the execution of the entire cut, while all other edges are weighted by the Euclidean distance between the corresponding points. The solution (see Fig. 5) consists of an ordered sequence of cutting line identifiers (odd values for the first point of the arrays, even values for the last), which is encoded in a `Int32MultiArray` message and published on the *sequence* topic.

Motion Planning

Once the trajectory and the corresponding sequence are received from the *trajectory* and *sequence* topics, respectively, the complete motion is generated starting from the current pose of the robot. From this initial configuration, a transition segment is first computed to connect the current robot position to the starting point of the first cutting line, as identified by the first element in the received sequence. The trajectory then proceeds along the selected cutting line, where the end-effector orientation is derived from the local geometry and filtered as discussed in Section II-A4. In the case of the jumper, after completing the cut, the end-effector is retracted from the garment of a predefined amount (0.05 m in the carried out test) and moved to the starting point of the next cutting line specified in the sequence, alternating between operational segments and transition segments. This process is repeated until all cutting operations have been executed, after which the robot is commanded to return to a predefined home position. The orientation of the end-effector during cutting is set to always face the centre of the object, forcing the robotic arm to track the path without colliding with the object.

Execution

Since the garments are not actually cut in these preliminary tests, the feedback and error-handling node is disabled. The execution node therefore retains only the functionality required to execute the received trajectory or target pose, which is forwarded to the external *MoveIt2* node for motion planning and control. In addition, the node continuously monitors the robot state to detect the end of the motion. Based on this information, it generates a Boolean signal to either trigger the next image acquisition step or terminate the process.

The main operations in the carried out tests are shown in the video available in [39] for both the jumper and the shoe.

IV. Conclusion

The industrial manufacturing sector is expanding, with increasing levels of automation across individual processes. However, small and medium-sized enterprises often face significant barriers, as current solutions typically rely on machines customized for specific garment types and operations. In this work, a reconfigurable robotic solution based on a ROS2 architecture has been proposed to support multiple operations on a wide range of apparel items and accessories. The main contributions of this study lie in the design of a unified perception–planning–execution pipeline, the capability

to adapt the processing strategy based on the garment type, and the integration of task sequencing and motion generation within a single framework. A case study dealing with a jumper and a shoe demonstrates the versatility of the approach across both planar and three-dimensional items, highlighting its potential for heterogeneous and unstructured scenarios. The achieved results, even if preliminary, suggest that the proposed system can reduce the need for multiple dedicated machines, thereby improving flexibility and cost-efficiency in industrial settings. The main limitation in the current experimental implementation is the lack of real cutting operations, together with a missing complete feedback loop for online adaptation. Future work will therefore focus on the integration of such feedback node, the adoption of a proper safe cutting tool, the management of motorized support systems, and the execution of complete closed-loop experiments. Additionally, parameter handling will be improved to further enhance system maintainability and scalability, for an easier deployment in real industrial environments.

REFERENCES

[1] H. Guo and C. Tsinopoulos, "Enabling a circular economy through green manufacturing in chinese apparel manufacturers: Antecedents and outcomes," *IEEE Transactions on Engineering Management*, vol. 71, pp. 9540–9554, 2024.

[2] K. A. Schumacher and A. L. Forster, "Textiles in a circular economy: An assessment of the current landscape, challenges, and opportunities in the united states," *Frontiers in Sustainability*, vol. Volume 3 - 2022, 2022.

[3] M. Shamsuzzaman, M. Islam, M. A. A. Mamun, R. Rayyaan, K. Sowrov, S. Islam, and A. S. M. Sayem, "Fashion and textile waste management in the circular economy: A systematic review," *Cleaner Waste Systems*, vol. 11, p. 100268, 2025.

[4] H. Jindal and S. Kaur, "Robotics and automation in textile industry," *International Journal of Scientific Research in Science, Engineering and Technology*, pp. 40–45, 05 2021.

[5] N. Singh, O. Singh, A. Singh, and V. Rengaraj, "Automation and robotics in apparel industry," *International Journal of Computer Applications Technology and Research*, vol. 13, pp. 24–32, 02 2024.

[6] Z. Paraskevi, "Robot handling fabrics towards sewing using computational intelligence methods," in *Robotic Systems - Applications, Control and Programming*, A. Dutta, Ed. London: IntechOpen, 2012, ch. 4.

[7] A. Ajith, G. Narayanan, J. Zornow, C. Calle, A. H. Lugo, J. L. S. Rincon, C. Wen, and E. Solowjow, "Robotic automation in apparel manufacturing: A novel approach to fabric handling and sewing," 2025.

[8] R. W. Kong, "Automated seam folding and sewing machine on pleated pants for apparel manufacturing," 2025.

[9] A. Longhini, M. C. Welle, I. Mitsioni, and D. Kragic, "Textile taxonomy and classification using pulling and twisting," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7564–7571.

[10] D. Seita, N. Jamali, M. Laskey, A. K. Tanwani, R. Berenstein, P. Baskaran, S. Iba, J. Canny, and K. Goldberg, "Deep transfer learning of pick points on fabric for robot bed-making," 2019.

[11] W. Yuan, Y. Mo, S. Wang, and E. Adelson, "Active clothing material perception using tactile sensing and deep learning," 2018.

[12] R. Kermenov, S. Foix, J. Borràs, V. Castorani, S. Longhi, and A. Bonci, "Automating the hand layup process: On the removal of protective films with collaborative robots," *Robotics and Computer-Integrated Manufacturing*, vol. 93, p. 102899, 2025.

[13] E. Corona, G. Alenyà, A. Gabas, and C. Torras, "Active garment recognition and target grasping point detection using deep learning," *Pattern Recognition*, vol. 74, pp. 629–641, 2018.

[14] K. Saxena and T. Shibata, "Garment recognition and grasping point detection for clothing assistance task using deep learning," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, 2019, pp. 632–637.

[15] A. Canberk, C. Chi, H. Ha, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Cloth funnels: Canonicalized-alignment for multi-purpose garment manipulation," 2022.

[16] R. Kermenov, A. Di Biase, I. Pellicani, S. Longhi, and A. Bonci, "Near time-optimal trajectories with iso standard constraints for human–robot collaboration in fabric co-transportation," *Robotics*, vol. 14, no. 2, 2025.

[17] H. Kim, W.-K. Jung, Y.-C. Park, J.-W. Lee, and S.-H. Ahn, "Broken stitch detection method for sewing operation using cnn feature map and image-processing techniques," *Expert Systems with Applications*, vol. 188, p. 116014, 2022.

[18] A. Bonci, A. D. Biase, S. Longhi, I. Pellicani, M. Prist, and A. Serafini, "Integrating llms into collaborative robotics for automated zipped apparel disassembly," in *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2025, pp. 1–8.

[19] D. Lopes, E. J. S. Pires, V. Filipe, M. F. Silva, and L. F. Rocha, "Intelligent and automated technologies for textile recycling pre-processing: A systematic literature review," *Technologies*, vol. 14, no. 4, 2026.

[20] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022.

[21] European Environment Agency, "Circularity of the EU textiles value chain in numbers," European Environment Agency, Report, 2024.

[22] European Commission, "EU strategy for sustainable and circular textiles," European Commission, Tech. Rep., 2022.

[23] S. Jain and V. Kumar, "Garment categorization using data mining techniques," *Symmetry*, vol. 12, no. 6, 2020.

[24] R. Tian, Z. Lv, Y. Fan, T. Wang, M. Sun, and Z. Xu, "Qualitative classification of waste garments for textile recycling based on machine vision and attention mechanisms," *Waste Management*, vol. 183, pp. 74–86, 2024.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[26] G. Csurka, R. Volpi, and B. Chidlovskii, "Semantic image segmentation: Two decades of research," 2023.

[27] H. M. Zakir and E. T. W. Ho, "Segment any class (sac): Multi-class few-shot semantic segmentation via class region proposals," 2024.

[28] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang, J. Lei, T. Ma, B. Guo, A. Kalla, M. Marks, J. Greer, M. Wang, P. Sun, R. Rädle, T. Afouras, E. Mavroudi, K. Xu, T.-H. Wu, Y. Zhou, L. Momeni, R. Hazra, S. Ding, S. Vaze, F. Porcher, F. Li, S. Li, A. Kamath, H. K. Cheng, P. Dollár, N. Ravi, K. Saenko, P. Zhang, and C. Feichtenhofer, "Sam 3: Segment anything with concepts," 2025. [Online]. Available: https://arxiv.org/abs/2511.16719

[29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[30] K. Guo, H. Ye, J. Gu, and H. Chen, "A novel method for intrinsic and extrinsic parameters estimation by solving perspective-three-point problem with known camera position," *Applied Sciences*, vol. 11, no. 13, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/13/6014

[31] A. Bonci, M. Pirani, and S. Longhi, "Robotics 4.0: Performance improvement made easy," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–8.

[32] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[33] R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, pp. 275–278, 1958.

[34] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM Review*, vol. 33, no. 1, pp. 60–100, 1991.

[35] L. F. Escudero, "The sequential ordering problem," *Annals of Operations Research*, 1999.

[36] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer, 2016.

[37] M. Ehrgott, *Multicriteria Optimization*. Springer, 2005.

[38] G. Nabissi, S. Longhi, and A. Bonci, "Ros-based condition monitoring architecture enabling automatic faults detection in industrial collaborative robots," *Applied Sciences*, vol. 13, no. 1, 2023.

[39] Video, "Video Demo for Framework's Experimental Testing," YouTube, April 2026, accessed: April 28, 2026. [Online]. Available: https://youtu.be/09RCQArJnzs