GNN Topology Representation Learning for Deformable Multi-Linear Objects Dual-Arm Robotic Manipulation

Alessio Caporali, Kevin Galassi, Riccardo Zanella, and Gianluca Palli, *Senior Member, IEEE*

***Abstract*—Deformable Multi-Linear Objects (DMLOs), or Branched Deformable Linear Objects (BDLOs), are flexible objects that possess a linear structure similar to DLOs but also feature branching or bifurcation points where the object's path diverges into multiple sections. The representation of complex DMLOs, such as wiring harnesses, poses significant challenges in various applications, including robotic systems' perception and manipulation planning. This paper proposes an approach to address the robust and efficient estimation of a topological representation for DMLOs leveraging a graph-based description of the scene obtained via graph neural networks. Starting from a binary mask of the scene, graph nodes are sampled along the objects' estimated centerlines. Then, a data-driven pipeline is employed to learn the assignment of graph edges between nodes and to characterize the node's type based on their local topology and orientation. Finally, by utilizing the learned information, a solver combines the predictions and generates a coherent representation of the objects in the scene. The approach is experimentally evaluated using a test set of complex real-world DMLOs. Within an offline evaluation, the proposed approach achieves a Dice score exceeding 90% in predicting graph edges. Similarly, the identification accuracy of *branch* and *intersection* points in the graph topology is above 90%. Additionally, the method demonstrates efficient performance, achieving a runtime of over 20 FPS. In an online assessment employing a dual-arm robotic setup, the approach is successfully applied to disentangle three automotive wiring harnesses, demonstrating the effectiveness of the proposed approach in a real-world scenario.**

***Note to Practitioners*—This research is motivated by the lack of effective robotic perception and manipulation solutions for deformable multi-linear objects such as wiring harnesses. These objects play a crucial role in several manufacturing industries. However, their intricate branching structures, inherent deformable nature, and limited distinctive visual characteristics present formidable challenges when automating tasks that involve them. The proposed approach addresses several of these challenges: 1) It maintains robustness in the presence of noise and disruptions in the input image mask due to its graph-based structure; 2) It remains unaffected by the existence of large connectors, clips, fixtures, and other objects along the branch sections of the object; 3) It avoids making any assumptions about the structure of the object; 4) It is capable of simultaneously identifying *branch-points* and resolving *intersection-points* among different branch *sections* of these objects. The experiments have verified the effectiveness of this solution when applied to real-world deformable multi-linear objects found in automotive settings. This underscores the potential to integrate this solution into industrial scenarios that involve the assembly and routing of such objects.**

***Index Terms*—State representation, graph neural network, deformable objects.**

Received 10 October 2024; revised 3 February 2025; accepted 15 April 2025. Date of publication 18 April 2025; date of current version 7 May 2025. This article was recommended for publication by Associate Editor G. Chen and Editor Z. Li upon evaluation of the reviewers' comments. This work was supported in part by European Commission's Horizon 2020 Framework Program with the Project REMODEL-Robotic Technologies for the Manipulation of Complex Deformable Linear Objects under Grant 870133; and in part by the Horizon Europe Project IntelliMan-AI-Powered Manipulation System for Advanced Robotic Service, Manufacturing and Prosthetics under Grant 101070136. The work of Alessio Caporali was supported by the Fondo Sociale Europeo Plus (FSE+) (2021–2027) under a research contract per Law 240/2010 (Art. 24(3)(a) and D.G.R. 693/2023, REF. PA: 2023-20090/RER) under Grant CUP: J19J23000730002. *(Corresponding author: Alessio Caporali.)*

Alessio Caporali, Kevin Galassi, and Gianluca Palli are with the Department of Electrical, Electronic and Information Engineering (DEI), University of Bologna, 40136 Bologna, Italy (e-mail: alessio.caporali@unibo.it).

Riccardo Zanella is with the Robotics and Mechatronics (RaM) Department, University of Twente, 7500 AE Enschede, The Netherlands.

This article has supplementary downloadable material available at https://doi.org/10.1109/TASE.2025.3562231, provided by the authors.

Digital Object Identifier 10.1109/TASE.2025.3562231

I. Introduction

NOWADAYS, the demand for increased automation in manufacturing processes is driving the development of new robotic systems capable of handling deformable objects effectively [1]. Deformable Linear Objects (DLOs), among the vast class of deformable objects, play a significant role in modern manufacturing processes, particularly in prominent sectors like automotive and aerospace [2], [3]. DLOs, like cables, wires, hoses, belts, and strings, are deformable structures that can change shape along a linear dimension. Unlike traditional solid objects, DLOs can bend and deform. Despite advancements in research concerning their perception [4], [5] and manipulation [6], [7], [8], relevant industrial tasks like cabling and wire routing still rely heavily on human intervention [2], [3], [9]. This highlights that significant challenges remain unresolved in this field.

Deformable Multi-Linear Objects (DMLOs), also known as branched DLOs [10], [11], are flexible objects that possess a linear structure similar to DLOs but also include branching or bifurcation points where the object's path splits into multiple branches. This added complexity in their deformation distinguishes them from simpler DLOs. Examples of such objects include wiring harnesses, as shown in Fig. 1, and branched hoses commonly found in manufacturing settings. Detecting and manipulating these objects require advanced techniques that can account for both their linear deformations and branching behavior. While several algorithms and

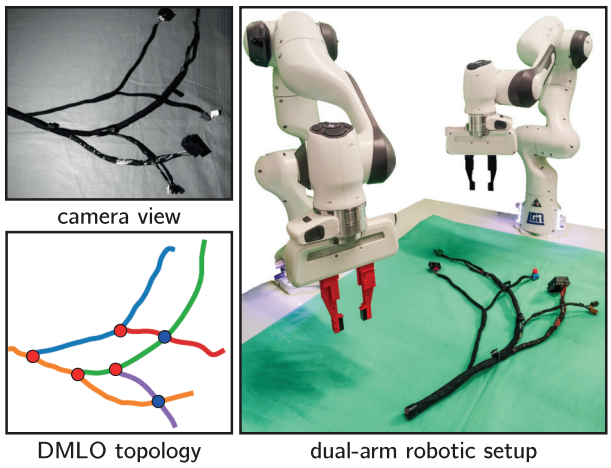


Fig. 1. Extracted topology representation of a deformable multi-linear object from a camera sample. The branch *sections* composing the object are displayed in different colors while the *branch-points* and *intersection-points* are highlighted with red and blue dots. On the right side, the dual-arm robotic experimental setup is shown.

approaches have been developed in recent years for general DLOs, addressing the perception and manipulation of DMLOs remains an ongoing challenge [2]. Indeed, existing works often rely on simplifying assumptions and specific scenarios [10], [11], [12], [13], [14], [15], [16], [17], [18]. For example, [11] proposes a topology matching approach that assumes the entire DMLO structure is fully visible and free from overlapping sections. Similarly, [10] addresses DMLO tracking under the assumption of knowledge of the DMLO model, allowing a physics simulator to run concurrently, with the initial scene requiring full visibility of the entire DMLO. Tracking is also addressed by [14], but the approach utilizes a simplified and unrealistic DMLO structure. In [12], DMLO perception is handled through a coarse localization method, which is challenging to apply for disentangling manipulation tasks. In contrast, our approach handles complex DMLO structures without relying on full visibility or idealized conditions. Importantly, we do not assume complete knowledge of the DMLO structure or simplified models. As output, our approach provides a complete topological description of the visible DMLO structure.

A. Contribution

In this paper, the challenging task of describing the configuration of a DMLO via a graph-based representation is tackled. This kind of representation is particularly suitable for DMLOs since it can be exploited easily in robotic manipulation tasks. The graph representation of DMLOs is derived through a multi-step process. Initially, graph nodes are extracted by sampling along the estimated centerlines of the objects on a binary mask of the scene. Subsequently, a data-driven pipeline is employed to acquire knowledge about how to assign graph edges between these nodes and determine the type of each node based on their local topology and orientation. Finally, by utilizing the learned information, a solver is applied to combine the predictions and generate a coherent representation of the objects in the scene.

Obtaining a robust and efficient graph representation of the DMLOs entails addressing two pivotal challenges: link prediction, which involves predicting the existence of connections between nodes in the graph, and node characterization, which aims to understand and capture key nodes' information such as their local orientation and topology inside the DMLO. These two challenges are tackled by exploiting Graph Neural Networks (GNNs) [19] to learn the topology of the DMLOs.

In particular, the GNN learning phase exploits an *inductive framework* [20]: The GNN weights are optimized based on a labeled training dataset while the model is deployed on unseen data (i.e. graphs) samples. For training, synthetic graphs obtained in a simulated environment are employed, allowing the optimization of the model on a large number of graphs and enabling important generalization capabilities in unseen real-world graphs.

The proposed method, thanks to the graph-based representation, provides robust predictions even for complex real-world DMLOs, i.e. it is not affected by the possible presence of large connectors, clips, and other fixtures along the object branch *sections*. Additionally, no prior knowledge of the DMLO structure is required, as the method relies solely on a binary mask of the scene. The method is capable of simultaneously identifying *branch-points* and resolving *intersection-points* among different branch *sections*.

The method is evaluated by performing a disentangling manipulative operation with a dual-arm robotic setup, shown in Fig. 1. The DMLOs employed in the experiments are complex real-world wiring harnesses from the automotive domain. The experiments allow us to demonstrate the effectiveness and accuracy of our framework in generating a robust and efficient topological representation leading to a simplified manipulation task by the robotic system.

To summarize, the main contributions of this paper are:

1) Description of the configuration of DMLOs with a topological graph obtained by exploiting a GNN model learned in an inductive framework, providing a robust and efficient representation;
2) Robustness to noise in the input mask and to the presence of connectors, clips, and other fixtures that can be connected to the branch *sections* of DMLOs;
3) Extensive evaluation in disentangling manipulative operations with complex real-world automotive wiring harnesses, demonstrating its effectiveness in generating an accurate and efficient topological representation, which simplifies manipulation tasks.

The associated data and method implementation are publicly available on https://github.com/lar-unibo/GNN_DMLO.

II. Related Works

A. Graph Neural Networks

A graph is a mathematical structure consisting of a set of nodes (or vertices) and a set of edges (also called links) that connect pairs of nodes [21]. Graphs are a powerful tool to represent data that can be modeled as a set of objects/points (nodes) and their relationships (edges). Indeed, graphs found applications in many fields, ranging from social networks to chemistry and biology [19].

Graph Neural Networks (GNNs) [22] are a powerful extension of classical graph analysis methods where the graph's

underlying structure is exploited in a learning framework. The idea of GNNs is to iteratively update node representations by aggregating information from their neighboring nodes of the graph. This allows GNNs to capture local relationships between the nodes, and through the propagation of the embedding via the network layers, it is also possible to capture more global relationships.

GNNs have been successfully applied to a wide range of tasks, including node classification, link prediction, and graph classification [19]. In particular, graph classification and link prediction are the two tasks also addressed in this paper. The first focuses on predicting the class or label of a set of nodes composing a graph. The task is usually performed by first learning node embeddings via a GNN, combining all the nodes embedding via a graph pooling operation, and then applying a classifier [19]. Link prediction aims to estimate the existence of connections between nodes in the graph. This task is usually performed by learning a similarity function between nodes and then predicting the existence of a link between two nodes based on their similarity [23].

Concerning deformable objects, GNNs were successfully adopted for the manipulation of linear and planar objects. In particular, GNNs are usually exploited to learn the interaction between key-points on clothes or ropes [24], [25], [26]. In [24], the GNN is combined with a recurrent neural network to track the key-points and a model predictive control scheme to plan the manipulation action. In [25] a second GNN is used also to plan the action. A GNN is employed as a forward dynamics module for a planar deformable object in [26].

In contrast to existing methods, our work is the first to leverage GNNs for topology estimation in DMLOs. GNNs are particularly effective at capturing the structural information inherent in graphs, utilizing node and edge features to develop robust representations that encode the relationships and dependencies within the data. This capability makes them well-suited for the task of learning the topology of DMLOs.

B. Deformable Objects State Representation

Deformable object perception and representation is still an open research problem [1]. Considering the complexity of the perception of such objects, many researchers use simplified scenarios to segment the objects from the scene, e.g. by using color difference, controlled backgrounds, or markers, as highlighted in [5]. This clearly limits the possible utilization of these methods in real scenarios, where the inconsistency of the background or the sensitivity to the quality of the segmentation mask can restrain a method from being used.

Learning-based methods have emerged as a key tool to tackle the perception problem of linear and planar deformable objects. For instance, key-points are automatically extracted from an image in [24] and [25].

Specifically for DLOs, several works have addressed the synthetic and real-world dataset generation [5], the instance segmentation task [4], [27], 3D shape estimation [28], [29], and crossing points detection [6]. Concerning DMLOs, a coarse localization method for branches is proposed in [12], highlighting the scarcity of solutions for addressing the complex perception and state representation challenges of DMLOs. Indeed, methods developed for DLOs can not be directly applied to DMLOs, but modifications and extensions [11] are required.

C. Deformable Multi-Linear Objects

Despite some research have been conducted in the past, e.g. wiring harness manufacturing automation in [15] and collaborative robots in wire harness assembly in [3], the state representation of DMLOs remains largely unexplored. Incorporating more structural information into the state representation of manipulated objects would greatly facilitate planning and manipulation tasks. For instance, [16] presents an automatic path-planning algorithm for wiring harness installations. However, this approach requires complete knowledge of the initial state and topology of the DMLO, which poses a significant limitation in real-world applications.

In an attempt to extract partial information, [17] developed a vision system that identifies the final connectors of a wire harness through image processing. However, they did not address other related tasks such as recognizing *branch-points* and *intersection-points*, leaving the geometric reconstruction of the main trunk and branch *sections* for future work. Another approach, presented in [12], involves classifying branch *sections* from raw RGBD images and augmenting the data using elastic transformation. However, this approach provides a black box solution encoding all information related to the branch *sections* and the DMLO into classification data, limiting its portability across tasks and/or different DMLOs.

Tracking of DMLOs has been investigated in several works [10], [14]. In [10] a modified registration method is combined with a topology model of the DMLO to track it under occlusions. The approach is evaluated on a wiring harness featuring connectors and clips, similar to the one employed in this work. However, knowledge of the DMLO structure is required and the DMLO is assumed to be in a fully visible and entirely disentangled configuration. Instead, in [14], tracking using depth images via a data-driven predictor is proposed. However, the method is evaluated on simplified DMLOs with no intersections between the branch *sections*, nor connectors and other disturbing objects along the branch *sections* themselves. Similarly, in [11], the matching between the observed and model DMLO is performed via a cost function. A score evaluating the topological uniqueness of a DMLO is also proposed.

The disentangling of DMLOs is investigated in [18], where a bin-picking policy to grasp and extract a DMLO in an untangled configuration is learned from real-world data.

Unlike existing approaches, our method does not require prior knowledge of the expected DMLO structure. Consequently, it can be applied to any DMLO, even if some parts are not visible. Additionally, it effectively handles overlapping *sections* by identifying specific *intersection-points*.

III. Method

The proposed method seeks to provide as output a topological representation of the DMLO, as shown in Fig. 2. The approach can be subdivided into three main stages, which are:

1) **Graph Initialization**: the graph representation is initialized from an image of the scene by sampling a set of nodes and edges;

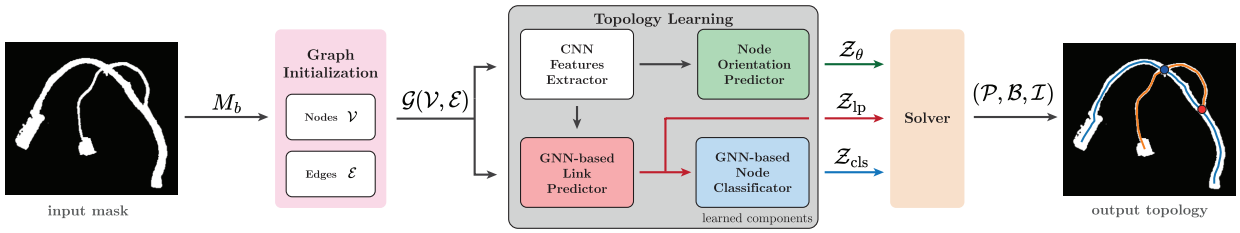


Fig. 2. Schema of the proposed approach with its main components: graph initialization, topology learning, and solver. On the right, the output topology representation is shown, where the individual branch sections of the DMLO are depicted in different colors, and the single *branch-point* and *intersection-point* are highlighted as red and blue dots, respectively.

2) **Topology Learning**: the graph edges are assigned a probability of existence; the nodes are classified according to their local topology and orientation;
3) **Solver**: the learned information is processed by a solver to obtain a coherent representation of the DMLOs in the scene.

In particular, the graph initialization phase is detailed in Sec. III-A. Then, the topology learning phase is described in Sec. III-B. The solver is described in Sec. III-C. The obtained representation can be used effectively to disentangle the DMLO branch *sections*, as explained in Sec. IV.

A. Graph Initialization

A graph with n nodes and m edges is initialized and represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each individual node i is denoted as v_i. Given two nodes v_i and v_j, their edge is indicated as e_{ij}. Since undirected edges are considered, $e_{ij} = e_{ji}$.

1) Semantic Segmentation: The input data to the proposed approach is a semantic segmentation binary mask M_b of the scene where the pixels are classified to either belong to DMLOs (value of 1) or not (value of 0). An example of a mask M_b is provided in Fig. 2.

Obtaining M_b is not the focus of this work as it is considered a pre-processing step. Indeed, the proposed approach is independent of the method used to obtain the semantic segmentation mask. Different approaches can be employed depending on application requirements, for instance deep-learning based approaches [5]. In this work, a calibrated 3D vision sensor is used to capture a point cloud of the scene. To extract M_b, a plane segmentation method is applied to the point cloud data, distinguishing points on the planar surface from those on the DMLO [30]. By projecting the DMLO points onto the image plane, M_b is generated. Although the extraction of the DMLO topology relies solely on M_b, the original 3D point cloud data is utilized for manipulation purposes, as discussed in Sec. IV.

2) Vertices Sampling: The set $\mathcal{V} = \{v_i\}_{i=1}^{n}$ contains the n vertices of the graph efficiently sampled from the binary mask M_b. The sampling strategy follows the approach originally presented in [4]. First, the distance transform operator is executed on M_b obtaining M_{dist}. This operator computes the Euclidean distances between the non-zero values of M_b and the nearest boundaries (zero/black values) [31], thus assigning an intensity value to each pixel based on the computed distance. In Fig. 3, M_{dist} originated from M_b (left side of Fig. 2) is shown where M_{dist} is color-mapped on the grayscale level from dark (zero distance) to bright (maximum distance).

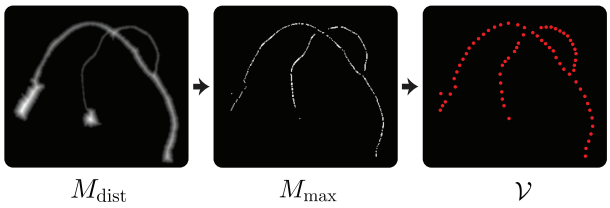


Fig. 3. Vertices sampling procedure from segmentation mask: distance transform maks M_{dist}, local maximum mask M_{max}, graph nodes $\mathcal{V}$.

Next, M_{dist} is dilated using a small square kernel, i.e. 3×3, resulting in M_{dil}. The local maxima of M_{dist} are then extracted by comparing pixel-wise values of M_{dist} and M_{dil}, masked using M_b. This process generates a binary map M_{max}, which indicates the positions of local maxima, approximating the center lines of the DMLOs in the mask. The computation of M_{max} can be expressed as follows:

$$M_{\text{max}}(i,j) = \begin{cases} 1 & if\, M_{\text{dil}}(i,j) = M_{\text{dist}}(i,j) \text{ and } M_b(i,j) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The set of maximum pixels of M_{max}, i.e. pixels whose value is equal to 1, is denoted as $\mathcal{V}_{\text{max}}$. The cardinality of $\mathcal{V}_{\text{max}}$ is quite high (see M_{max} in Fig. 3). Thus, the farthest point sampling algorithm [32] is employed to down-sample $\mathcal{V}_{\text{max}}$. The ratio $\alpha \in [0, 1]$ is used to specify the amount of down-sampling. It is chosen such that the density of the nodes is kept mostly constant. This is important since different DMLOs can have very diverse amounts of branch *sections* and thus foreground pixels. The constant density is obtained by dividing the current mean point-to-point distance in $\mathcal{V}_{\text{max}}$ with the target distance to achieve. Therefore, the set of vertices $\mathcal{V}$ of the graph $\mathcal{G}$ is obtained as $\alpha\mathcal{V}_{\text{max}}$.

This vertices sampling process ensures an efficient representation of the DMLO's topology, reducing redundancy while preserving the key structural information necessary for subsequent analysis and modeling.

3) Edges Sampling: In order to create a set of initial edges, referred as $\mathcal{E}_{\text{knn}}$, the proximity between vertices is leveraged. This involves retrieving and using the K_{nn} nearest neighbors in $\mathcal{V}$ as edges for each vertex. This work employs a value of 8 for K_{nn}. By utilizing a sufficiently large number of nearest neighbors as initial edges, comprehensive coverage of the local connectivity in the graph is guaranteed, which can be advantageous for the subsequent learning phases of Sec. III-B.

B. Topology Learning

Learning on graphs using GNNs involves processing the graph structure, node features, and edge features [19].

In a graph with n nodes, the graph structure can be represented by an adjacency matrix $A \in \mathbb{R}^{n \times n}$. The adjacency matrix A encodes the connections between nodes, where $A_{ij} = 1$ if there is an edge between nodes i and j, and $A_{ij} = 0$ otherwise.

The node features in the graph are represented by a node feature matrix identified as $F_{\mathrm{v}} \in \mathbb{R}^{n \times h}$, where each row i corresponds to the h features associated with node n_i. The edge feature matrix, denoted as $F_{\mathrm{e}} \in \mathbb{R}^{m \times d}$, represents instead the features associated with each edge in the graph. Here, d denotes the dimensionality of the edge features.

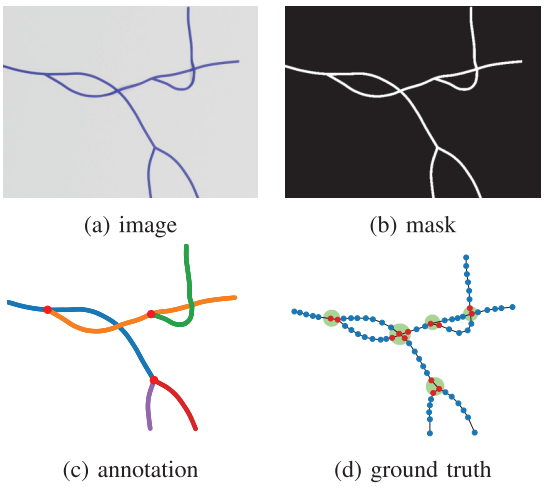
By incorporating the adjacency matrix A, the node feature matrix F_{v}, and the edge feature matrix F_{e}, GNNs can effectively process and learn from the structural, node, and edge information of the graph to perform various graph-based learning tasks.

In the context of this paper, the graph learning strategy is based on an *inductive framework* with supervision provided by ground truth data [20]. A synthetic dataset of DMLO graphs is generated and exploited, as detailed in Sec. III-B1. The nodes and edges feature computations are discussed in Sec. III-B2. Hence, the link and node orientation prediction tasks are presented in Secs. III-B3 and III-B4. In Sec. III-B5 the node subgraph classification task is analyzed and Sec. III-B6 details the overall training strategy.

1) Synthetic Dataset: To train the GNN model, a synthetic dataset is exploited. Similarly to [5], each *section* of the synthetic DMLO is modeled as a 3rd-order spline curve and a rendering pipeline based on Blender [33] is used to obtain the image samples.

First, a synthetic DMLO in a random configuration is generated. It is composed of a maximum of 5 main *sections*, organized by up to three *branch-points*. A *branch-point* is defined as an area of the DMLO where two different *sections* are connected. Then, the camera location is randomized and an image of the scene is rendered. In Fig. 4a an example of the generated data is provided.

Along with the generated image, the randomly generated spline curves are saved and used for annotation purposes. Indeed, since the proposed learning strategy is supervised, for each generated mask sample (Fig. 4b), annotation data is required. Therefore, a ground truth graph is generated based on the available annotations. The vertices of the ground truth graph are obtained by exploiting the sampling strategy of Sec. III-A2. The edges are instead computed by processing the ground truth spline curve used to synthesize the DMLO sample. These curves are projected from the cartesian space to the image plane by knowing the camera location in the synthetic scene, as shown in Fig. 4c. Then, each sampled node is associated with the correct curve. The ordering of the vertices and thus the sequence of edges to add is obtained by traversing the vertices on the corresponding ground truth curve. The only edges missing are the ones at the *branch-point* locations. In this case, since the structure of the synthetic DMLO is known, proper edges are added. The result is a full graph representation of the scene provided by the mask, as shown in Fig. 4d. The introduced ground truth edges are exploited in Sec. III-B3.



(a) image (b) mask (c) annotation (d) ground truth

Fig. 4. Sample of the synthetic dataset. In (a) and (b) the rendered image and mask of the randomly generated DMLO. In (c), the colored curves represent the different sections and the red dots describe the *branch-points* used in the annotation process. In (d) the obtained ground truth graph is displayed with normal (blue) and highly-connected (red) node classes.

Building on the association between the graph nodes and the initial spline curves, a ground truth value for each node's direction can be established by taking the tangent of the spline at the node's location. This directional information is utilized in Sec. III-B4. Additionally, the spline curves help identify *branch* and *intersection* points. A region of high connectivity is characterized by the local thickness (highlighted in green in Fig. 4d), which labels nodes as either highly-connected (i.e., those near *branch* and *intersection* points) or normal. This classification is later exploited in Sec. III-B5.

2) Encoding Nodes and Edges: Node and edge features play a critical role in enhancing GNN performance [19]. Each node is assigned an input feature vector derived from two sources: (1) the encoded, normalized absolute position of the node in the image plane and (2) values sampled from the encoded mask M_{enc} at the node's location.

a) Encoding of Node Positions: The normalized node positions in the $(0, 1)$ range are encoded using a linear layer with ReLU activation, mapping 2D pixel locations to a h-dimensional space, resulting in $F_{\mathrm{P}} \in \mathbb{R}^{n \times h}$. Each row of F_{P} represents a node's encoded positional feature.

b) Encoding of Mask: For the mask encoding, the M_{dist} mask is first normalized and then processed through a two-layer CNN. Each convolutional layer uses a 5×5 kernel with a padding of 2 to maintain the input image size. A batch normalization layer is included between the convolutional layers to stabilize the training process. Finally, a max pooling operation is performed with a 2×2 kernel and a stride of 2. The first convolutional layer expands the input channels from a dimension of 1 to $h/2$, while the second layer increases the channel's dimension further to h. This encodes the M_{dist} mask into a new mask M_{enc} with half the dimensions and h channels. The node features are then extracted from M_{enc} by sampling each node location (adjusted by the new dimensions) along all the h channels. This results in the matrix F_{M} of size $\mathbb{R}^{n \times h}$ where each row encodes the mask information at each node location.

The encodings obtained from F_{P} and F_{M} are concatenated column-wise, resulting in the combined node feature matrix

$F_{\text{PM}} \in \mathbb{R}^{n\times 2h}$. Each row of F_{PM} represents a node's feature vector, combining position and encoded mask information. A final non-linear transformation is performed on F_{PM} using a linear layer with ReLU activation to down-sample the feature dimension to h. The output of this transformation is the final node feature matrix $F_{\text{v}} \in \mathbb{R}^{n\times h}$.

c) Encoding of Edges: Edge features are crucial for GNN performance and are derived from the edge direction and norm. In particular, for each edge in the set $\mathcal{E}_{\text{knn}}$, the norm is computed, normalized, and encoded via a linear layer from a dimension of 1 to $h/2$ followed by a ReLU activation function. Likewise, the edge direction is converted to an angular value and first encoded as a 180-dimensional vector with entries defined by applying a Gaussian function centered at the edge angle. This encoding is necessary to address the angular periodicity [34]. Then, the 180-dimensional vector is embedded by a linear layer from a dimension of 180 to $h/2$ followed by a ReLU activation function. The embedded norm and direction are concatenated and processed by another linear layer, yielding an edge feature matrix $Fe \in \mathbb{R}^{m\times h}$, where each row represents an edge feature vector.

3) Link Prediction: Link prediction in graph theory refers to predicting the likelihood of an edge (link) between two nodes. In this paper, the task is to determine the probability of existence for each edge in the set $\mathcal{E}_{\text{knn}}$.

To accomplish this, a Graph AutoEncoder (GAE) [35] is utilized, which is characterized by an encoder-decoder structure. The encoder takes the initialized graph from Sec. III-A as input and uses GNN layers to aggregate and update the features of each node based on the extracted input features of Sec. III-B2. The encoder's purpose is to learn a latent representation that captures the underlying structure and patterns in the graph.

On the other hand, the decoder processes the node features obtained from the encoder through multiple neural network layers. When given two nodes to test, the decoder produces a value that represents the probability of their connection. The decoder's aim is to rebuild the graph structure based on the learned node representations.

a) GNN Structure: To perform link prediction using GNNs, a 3-layer GNN is utilized to process the node and edge features along the graph structure. Each GNN layer aggregates and updates the node features followed by a non-linear transformation. The GNN layer operation, adapted from [36], can be expressed as follows:

$$x_i^{(k+1)} = \sigma\left(\sum_{u\in N(v_i)} (W x_u^{(k)} + B x_i^{(k)} + K f_e^{(u,i)}) + S\, x_i^{(k)}\right) \tag{2}$$

Here, $x_i^{(k+1)}$ represents the updated embedding vector of node v_i in the $(k+1)$-th GNN layer. The terms W, B, and K are trainable matrices that perform linear transformations of the neighbor embeddings, i.e. $u \in N(v_i)$, the current node embedding x_i^k, and the edge feature between the considered nodes $f_e^{(u,i)}$, respectively. The matrix S is a trainable matrix that accounts for a skip connection. The function σ represents a non-linear activation function, such as ReLU. Each GNN layer transforms the node features while maintaining their dimensions as h. Two batch normalization layers are placed between the GNN layers to help stabilize the training process. For the first GNN layer, the embedding x_i^0 corresponds to the node features row-vector f_v^i extracted from matrix F_v.

b) Links Decoding: Following the GNN processing, the node embeddings obtained by the last layer are used to predict the probability of a connection for each edge in the set $\mathcal{E}_{\text{knn}}$. The link predictor decoder is conceived as a 3-layer MLP (Multi-Layer Perceptron) with an input and output dimension of h and ReLU activation functions between the layers, except for the last layer. Given two graph nodes v_i and v_j with their encoded node features x_i and x_j, the probability of a connection between them is obtained as:

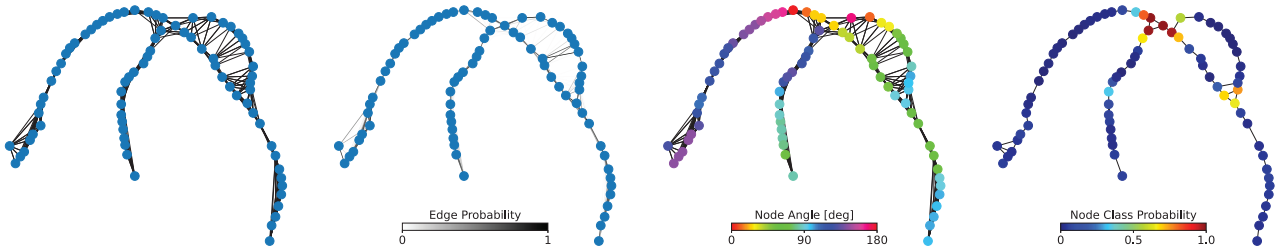
$$z_{ij}^{lp} = \sigma(\text{MLP}(x_i \cdot x_j)) \tag{3}$$

Here, σ represents a sigmoid activation function, ensuring that the predicted probability is within the range (0, 1). The MLP applies a series of linear transformations and non-linear activations to the dot product of the node features x_i and x_j before producing the final probability value. In Fig. 5a the link probabilities predicted for a sample input graph (Fig. 5b) are shown.

4) Node Orientation Prediction: When representing linear objects and shapes, it is possible to perform an orientation characterization for each vertex of the graph. This characterization aims to describe the local section of the linear object near the vertex by assigning an orientation attribute to the vertex itself. To achieve this, the features F_{M} extracted in Sec. III-B2 with dimensions h are mapped to a 180-dimensional vector through a single neural network linear layer followed by a softmax activation function. This process is similar to what is done in [4], with the fundamental difference being that the input features are extracted from the encoded mask M_{enc} instead of employing fixed-size crop images of M_{dist}. The resulting 180-dimensional vector represents the node orientation as a Gaussian function centered at the predicted orientation, similar to the encoding performed to the edges in F_{e} (see Sec. III-B2). To obtain the actual angular value, the argmax function is applied to determine the peak of the predicted distribution. An example of the estimated node orientations for a sample graph is shown in Fig. 5c.

5) Node Subgraph Classification: Until now all the nodes $v \in \mathcal{V}$ are regarded as similar with no distinction. However, in the context of DMLOs, it is possible to distinguish between two types of nodes: highly-connected nodes and normal nodes. A highly-connected node is defined as a node having a degree of more than 2, which instead is the limit for normal nodes, i.e. a node that is contained in a given *section* of the DMLO. Highly-connected nodes are *branch-points*, where two *sections* are connected, and *intersection-points*, where two *sections* are overlapping.

The node classification task is employed to predict the type of each node in the graph among the two possible ones. Classifying directly the nodes of the graph is challenging, even exploiting the graph structure, since the condition of being a highly-connected node is due to an area around the node rather than the node itself. To better characterize the task, subgraphs of one hop around the considered node are computed and used in the classification task. The features of the nodes composing the subgraph are exploited to enrich the set of data making the classification problem easier. A GNN is utilized to process the node features of the subgraph and predict the probability of



(a) initial edges $\mathcal{E}_{\text{knn}}$ (b) link probabilities $\mathcal{Z}_{\text{lp}}$ (c) node orientations $\mathcal{Z}_{\theta}$ (d) node class probabilities $\mathcal{Z}_{\text{cls}}$

Fig. 5. Outcomes of the different stages of the pipeline. From the graph with the input set of edges $\mathcal{E}_{\text{knn}}$ (a), the predicted edge probabilities are displayed in (b) while the node orientations are in (c). The result of the node subgraphs classification task between *normal* (0) and *highly-connected* (1) nodes is shown in (d).

the entire subgraph originating from a highly-connected (label 1) or normal (label 0) node.

As node features, the encoded position and mask information are exploited as described in Sec. III-B2. In addition, the node embedding obtained by the last GNN encoder layer of Sec. III-B3 is concatenated as well. Thus, a linear layer is used to process the node feature vector of dimension $3h/2$ to a dimension of h.

As edge features, the encoded norms and direction as detailed in Sec. III-B2 are exploited. The output of the decoder of Sec. III-B3 is processed by a linear layer with input dimension 1 and output dimension h followed by a ReLU activation function, obtaining F_{e}^{lp}. The final edge features are obtained by summing the entries of F_{e} and F_{e}^{lp}.

The structure of the GNN is similar to the one employed in Sec. III-B3. It is composed of one GNN layer with an input and output dimension of h and a ReLU activation function. Then, as output, an add-pooling operation is performed at the graph level to aggregate all the features of the nodes into a single embedding vector of dimension h. Finally, two linear layers compress the embedding first to a dimension of $h/2$ and then to an output dimension of 1. A sigmoid activation function is thus applied to the output to obtain the final probability value in the range (0-(1). In Fig. 5d an illustration describing the predicted node classes for a sample graph is depicted.

6) Training Strategy: For all network training, the binary cross-entropy loss function is employed. The link prediction task of Sec. III-B3 and the node orientation prediction task of Sec. III-B4 are optimized together with a combined loss function, as:

$$\mathcal{L}_{\text{link},\theta} = \mathcal{L}_{\text{link}} + \mathcal{L}_{\theta} \tag{4}$$

The link prediction task is optimized by computing the loss among the positive (predicted value of 1) and negative (predicted value of 0) edges. The first relates to the edges present in the ground truth graph as described in Sec. III-B1. The latter is obtained starting from the set $\mathcal{E}_{\text{knn}}$ by subtracting the ground truth edges.

Regarding the node orientation prediction task, the smooth label strategy is employed to map the angular values from the $[0°, 180°]$ domain to a 180-dimensional domain in the form of a Gaussian distribution centered at the angular value [34]. Thus, this encoding is applied to the angular ground truth data of Sec. III-B1 and the network is trained to predict a similar output in the form of a classification task with respect to the 180-dim classes, i.e. the node orientation angles in degrees. The binary-cross entropy loss $\mathcal{L}_{\theta}$ is thus computed element-wise on the predicted and ground truth value across the vector dimension.

The node subgraph classification network of Sec. III-B5 is also trained with a binary-cross entropy loss between the node labels and the output provided by the network. The labels are obtained from the ground truth graph, where nodes in the proximity of *branch-points* and *intersection-points* locations are given a value of 1 instead of 0 assigned to all the others. Since the ratio between the 0-class and 1-class is quite unbalanced toward the first one, the 0-class is downsampled to roughly match the number of elements in the 1-class.

C. Solver

To compute a final representation of the DMLO, the solver combines predicted link probabilities, node orientations, and node classes. The solver, outlined in Alg. 1, performs three main steps:

1) Edges filtering: the input graph edges are refined based on the predicted link probabilities (Sec. III-B3) and nodes orientations (Sec. III-B4) as described in Sec. III-C1;
2) High-degree nodes handling: based on the predicted node classes, high-degree nodes are interpreted and handled accordingly, as detailed in Sec. III-C2;
3) *Sections*, *branch-points* and *intersection-points* extraction: the final topology representation is computed, as detailed in Sec. III-C3.

In the following, each step is discussed in detail.

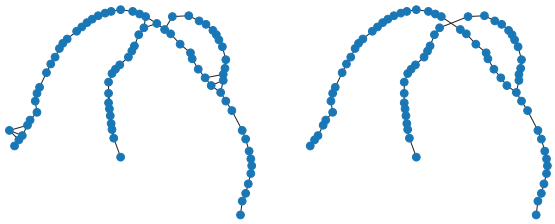
1) Edges Filtering: The first step of the solver is filtering the edges of the input graph. Indeed, the cardinality of $\mathcal{E}_{\text{knn}}$ is quite large since the majority of edges are either connecting two different DMLO *sections* or just redundant within the same *section*. The filtering is accomplished by first accounting the predicted link probabilities $\mathcal{Z}_{\text{lp}}$ (line 1), and then by considering also the node orientations $\mathcal{Z}_{\theta}$ (line 1).

a) Link Prediction Solver: It takes into account the predicted link probabilities $\mathcal{Z}_{\text{lp}}$ to select the edges that are more likely to be part of the same DMLO *section*. The link probabilities are computed by the GNN network of Sec. III-B3. The link prediction solver is employed to just remove all the very unlikely edges in the graph, thus a quite low threshold value (e.g. 0.05) on the computed probabilities is employed and only the edges having a probability greater than the threshold are kept. The resulting graph is $\mathcal{G}_{\text{lp}} = (\mathcal{V}, \mathcal{E}_{\text{lp}})$.

Algorithm 1 Solver

Input: $\mathcal{V}$, $\mathcal{E}_{\text{knn}}$, $\mathcal{Z}_{\text{lp}}$, $\mathcal{Z}_{\theta}$, $\mathcal{Z}_{\text{cls}}$
Output: $\mathcal{P}$, $\mathcal{B}$, $\mathcal{I}$

/* **Edges Filtering** */
$\mathcal{E}_{\text{lp}} \leftarrow$link_prediction_solver($\mathcal{V}$, $\mathcal{E}_{\text{knn}}$, $\mathcal{Z}_{\text{lp}}$)
$\mathcal{E}_{\text{f}} \leftarrow$ edges_simplification($\mathcal{V}$, $\mathcal{E}_{\text{lp}}$, $\mathcal{Z}_{\theta}$)
$\mathcal{G}' \leftarrow (\mathcal{V}, \mathcal{E}_f)$
/* **High-degree Nodes Handling** */
$\mathcal{E}_0 \leftarrow$ process_normal_nodes($\mathcal{G}'$, $\mathcal{Z}_{\text{cls}}$)
$\mathcal{E}_1 \leftarrow$ process_high_deg_nodes($\mathcal{V}$, $\mathcal{E}_0$, $\mathcal{Z}_{\text{cls}}$)
$\mathcal{G}'' \leftarrow (\mathcal{V}, \mathcal{E}_1)$
/* **Topology Output** */
$\mathcal{B} \leftarrow$ branch_points_extraction($\mathcal{G}''$)
$\mathcal{P} \leftarrow$ sections_extraction($\mathcal{G}''$)
$\mathcal{I} \leftarrow$ intersection_points_extraction($\mathcal{G}''$)
return $\mathcal{P}$, $\mathcal{B}$, $\mathcal{I}$



(a) $\mathcal{G}'$ (b) $\mathcal{G}''$

Fig. 6. Effect of solver stages: (a) Obtained graph after the application of the edge filtering stage on the input graph of Fig. 5a; (b) The result of the high-degree nodes handling exploiting the estimated node classes.

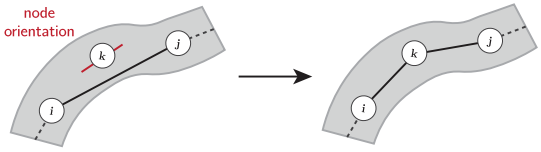
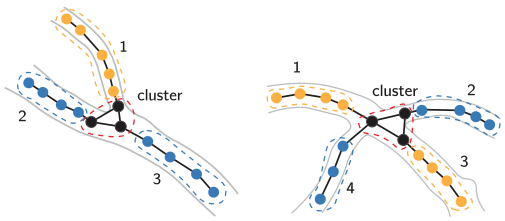


Fig. 7. Node orientation solver updating the edges around node k in case its orientation is consistent with the nearby existing edge e_{ij} between nodes i and j.

b) Node Orientation Solver: This step is performed to remove redundant edges that result, for instance, in graph cycles. The edges are checked to remove the ones that are mostly aligned with other nodes. The alignment is evaluated by considering the node orientations $\mathcal{Z}_{\theta}$. In particular, the orientation of a node is an angular value, which is converted into a direction vector, e.g a direction vector of node v_i with orientation z_i^{θ} is obtained as $d_i = \left[\cos(z_i^{\theta})\ \sin(z_i^{\theta})\right]^{\top}$. The cosine similarity is employed to evaluate the alignment between the direction vectors of the edges and the nodes. The cosine similarity for two general vectors a and b is defined as

$$\cos\_\text{sim}(a, b) = \frac{a^T b}{\|a\| b}. \tag{5}$$

With reference to Fig. 7, let's consider one edge e_{ij} and a separate node v_k. The direction of edge e_{ij} is denoted as d_{ij}^e. The distance between the edge segment and node v_k is computed. If the distance is less than the estimated DMLO thickness at point v_k (this is easily obtained by M_{dist}), then the edge is removed if the orientation of node v_k is consistent



(a) (b)

Fig. 8. Examples of cluster formation in regions containing high-degree nodes, indicated in black. The numbers are used to indicate the sub-*sections*.

with the one of edge e_{ij}, i.e. $|\cos\_\text{sim}(d_k, d_{ij}^e)| < 0.5$ where $|\cdot|$ denotes the absolute value. In particular, two additional edges e_{ik} and e_{kj} are inserted to compensate for the removal of e_{ij} as shown in Fig. 7. This process can be applied to multiple nodes simultaneously by projecting them onto the edge and placing new edges accordingly. Moreover, it is carried out for all nodes in $\mathcal{V}$ via matrix multiplications, resulting in efficient processing times. All the selected edges are stored in $\mathcal{E}_{\text{f}}$ obtaining the graph $\mathcal{G}'$ as shown in Fig. 6.

2) High-Degree Nodes Handling: The predicted node classes $\mathcal{Z}_{\text{cls}}$ of Sec. III-B5 are exploited to further refine the graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}_{\text{f}})$. First, the predicted *normal* nodes are processed followed by the high-degree nodes clustering and processing. Considering the predicted values $\mathcal{Z}_{\text{cls}}$, the nodes in $\mathcal{V}$ with a class score below a threshold value (e.g. 0.5) are considered *normal* nodes, $\mathcal{V}_{\text{normal}}$. The remaining nodes are considered *high-degree* nodes, $\mathcal{V}_{\text{high}}$.

a) Normal Nodes Processing: Among the *normal* nodes, those having a node degree higher than 2 in $\mathcal{G}'$ are processed in order to simplify the graph since their connectivity is in contrast with the predicted class. For a given node v_i characterized by a degree larger than 2, all the edges connected to v_i are retrieved and organized in combinations of two elements. Indeed, since v_i is assumed to be part of a DMLO *section*, it should have just two edges. Among the combinations, the one describing the smoothest connection is selected and the other edges are removed. The smoothness is evaluated by employing the cosine similarity as in eq. (5). In particular, considering two edges e_{ji} and e_{ik} of node v_i, their smoothness is obtained as $s_{jik} = \cos\_\text{sim}(d_{ji}^e, d_{ik}^e)$. The final set of edges $\mathcal{E}_0$ is computed after processing all the *normal* nodes.

b) Cluster-Based High-Degree Nodes Processing: The nodes in $\mathcal{V}_{\text{high}}$ are processed by first organizing them into *clusters* considering as graph $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E}_0)$. Each cluster describes one given DMLO area characterized by high-degree nodes. Indeed, since multiple nodes can describe the same *branch-points* and *intersection-points* area of the DMLO, neighboring nodes will have similar class predictions, as shown in Fig. 5d where the two high-degree areas of the graph show smooth values between 0 and 1. Thus, the clustering is performed by collecting the neighbors of each node in $\mathcal{V}_{\text{high}}$ and accounting for possible overlapping. Fig. 8 shows examples of clusters surrounded by sub-*sections* (e.g. 1, 2 and 3 in Fig. 8a).

Then, each cluster is solved by connecting the individual sub-*sections* in the most smooth way possible. For instance, as illustrated in Fig. 8, which shows the clustering results for two different examples, the following connections are established: in Fig. 8a, sub-*sections* 2 and 3 are connected, while

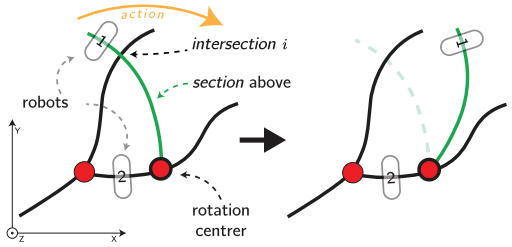


Fig. 9. Schematic of the manipulation motion originated for the represented topology.

in Fig. 8b, the connections are made between sub-*sections* 1-3 and 2-4. The merging process involves evaluating proposals for merging sections and calculating a smoothness score, similar to the approach applied at the node level using eq. 5.

The final set of edges, $\mathcal{E}1$, is thus determined, and the resulting graph $\mathcal{G}'' = (\mathcal{V}, \mathcal{E}_{\text{int}})$ is constructed. An example of the computed graph $\mathcal{G}''$ is shown in Fig. 6b.

3) Sections, Branch- and Intersection-Points Extraction: The last step of the solver is the computation of the final topology representation. Only at this stage, a key assumption is leveraged to classify *branch-* and *intersection-points* based on the degree of the nodes in $\mathcal{G}''$. In particular, nodes with a degree of 3 are classified as *branch-points* whereas nodes with a degree of more than 3 as *intersection-points*. Therefore, *branch-points* are extracted from the graph $\mathcal{G}''$ and stored in $\mathcal{B}$ (line 1 in Alg. 1). Then, the *sections* are computed by traversing the graph starting from the nodes with degree 1 or degree 3, and ending at the nodes with degree 1 or degree 3. The *sections* are stored in $\mathcal{P}$ (line 1). Finally, the *intersection-points* are obtained by checking the intersections between different *sections* in $\mathcal{P}$ and saving the result in $\mathcal{I}$ (line 1).

IV. Topology-Driven Manipulation

The effectiveness of the proposed DMLO topology representation is demonstrated through a manipulation task. This section introduces the manipulation framework, focusing on the problem of spreading (or disentangling) a DMLO by removing all *intersections* between its *sections*.

In this context, the following three assumptions are introduced: 1) the structure of the DMLO is not known, therefore no information from it is exploited, e.g. root location. 2) Given the stiffness of the DMLO considered, e.g. an automotive wiring harness, the motion of a section of the DMLO is approximated by a rigid transformation constrained to rotate around the associated *branch-point*; 3) Although the method of Sec. III does not require 3D data, for the following disentangling task, we assume that depth information is available. This allows for 1) transforming the DMLO topology from the image coordinate system to the robot's Cartesian coordinate system, and 2) simplifying the identification of the above *section* for DMLO disentanglement.

To solve the manipulation task, a dual-arm robot system is employed. The first arm (robot 1) performs a pick-and-place operation with a motion trajectory around an estimated center of rotation, i.e. the *branch-point*. The second arm (robot 2) is used to keep a part of the DMLO fixed in place. An illustration is provided in Fig. 9.

The manipulation task is detailed in Alg. 2. As input, the topology representation of the DMLO contained in the scene under analysis is provided, in the form of *sections* $\mathcal{P}$, *branch-points* $\mathcal{B}$ and *intersection-points* $\mathcal{I}$. The goal is to solve all the possible intersections present in the DMLO (line 2), where each possible manipulation action computed is saved into $\mathcal{A}$ (line 2).

For a given intersection i, the segments characterizing it are retrieved at line 2. Given the two segments, the *section* of the DMLO above at the intersection i is evaluated from the 3D data (line 2) by selecting either $s_{i,0}$ or $s_{i,1}$. The chosen *section* is the one whose nodes closest to the intersection point have the largest average distance along the z-axis from the underneath plane. Therefore, the *branch-point* constraining the motion of s_{above} is retrieved as the one closer to the intersection i along s_{above}, in the direction of its encounter with the other section (line 2). This will give the section to move and the rotation center of the movement described in Fig. 9.

The pick-and-place trajectories (line 2) are generated by simulating a circular motion of s_{above} around the fixed center (held by the other arm). The rotation direction is determined by the angle increase at the *branch-point*, i.e. spreading objective. The designated grasp locations are strategically positioned just before the rotation-center *branch-point* and just after the *intersection-point*, see Fig. 9. This placement is crucial for minimizing the risk of collisions with other parts of the object. Given multiple *intersection-points*, the one to manipulate is selected as the most external in the graph topology (line 2), i.e. closer to graph terminal nodes with degree 1, since it is assumed to be easier to solve. Fig. 9 illustrates the approach. Task assignment for each arm (hold or motion) is based on the distance to the target poses.

Algorithm 2 DMLO Intersection Manipulation

Input: $\mathcal{P}$, $\mathcal{B}$, $\mathcal{I}$
Output: dual-arm manipulation to perform

1 $\mathcal{A} \leftarrow 0$
2 **for** $i \in \mathcal{I}$ **do**
3 $\quad s_{i,0}, s_{i,1} \leftarrow$ GetSegmentIntersection(i)
4 $\quad s_{\text{above}} \leftarrow$ getSegmentAbove($s_{i,0}, s_{i,1}$)
5 $\quad b \leftarrow$ getBranchPoint (s_{above})
6 $\quad \mathcal{T}_1, \mathcal{T}_2 \leftarrow$ getPickAndPlaceTrajectories(b, s_{above})
7 $\quad \mathcal{A} \leftarrow [\mathcal{T}_1, \mathcal{T}_2]$
8 $a^* \leftarrow$ getBestAction(A)
9 dual_arm_manipulation(a^*)

The proposed method is just one possible approach to the disentangling task, and we acknowledge that more effective methods may exist. However, the goal of this section is to present a straightforward, easy-to-implement manipulation approach that leverages the estimated topology from Sec. III for deployment in a robotic task.

V. Experiments

The experiments involve the evaluation of the proposed topology representation pipeline in a real-world manipulation scenario. The experimental setup is composed of a robotic cell containing two Panda robots from Franka Emika shown in Fig. 1. The robotic cell is equipped with a Photoneo MotionCam3D structured-light camera statically mounted.

(a) *WH1* (b) *WH2* (c) *WH3*

Fig. 10. DMLOs real-world test samples.

Details about the training dataset and hyperparameters are provided in Sec. V-A whereas the DMLOs employed in the experiments are presented in Sec. V-B. Then, Sec. V-C discusses the manipulation experiment that is used to both evaluate the application of the proposed method for manipulation purposes and also to collect test samples. Sec. V-D analyses the collected samples. An analysis of the timing performances of the method is provided in Sec. V-E. Comparisons with methods from the DLOs domain are performed in Sec. V-F. The limitations are detailed in Sec. V-G.

A. Training Details

The synthetic dataset introduced in Sec. III-B1 is exploited in the training phase. It consists of a total of about 1600 graph samples used for train and 280 graph samples used for validation. First, the link predictor and node orientation predictor are trained. A learning rate of 10^{-3} is employed with the Adam optimizer. The batch size is set to 2, the hidden dimension h to 32, and the training is performed for 200,000 steps. The best model is selected based on the validation loss. Then, the node subgraph classificator is trained by employing the same dataset but also the previously trained predictors, which are kept frozen in this last training phase. A similar training process is employed. In particular, a learning rate of 10^{-5} is used with the Adam optimizer. The batch size is set to 64, the hidden dimension h to 32, and the training is performed for 50,000 steps. The best model is selected based on the validation loss.

B. Test DMLOs

The experiments are performed using real-world automotive wiring harnesses, shown in Fig. 10. Samples differ by increasing the complexity of the structures and the number of final endpoints. The samples, with reference to Fig. 10, are named and characterized in the following way:

- WH1: Plain taped cables with 1 *branch-points* and two end-points.
- WH2: Taped cables with side routing clips. A total of 3 *branch-points* and 5 end-points. Various final connectors with different sizes and shapes.
- WH3: Taped cables with side routing clips. A total of 5 *branch-points* with multiple end-points of different sizes.

C. Manipulation Experiment

The manipulation experiment is used to evaluate the performance of the proposed pipeline in a real-world scenario and to collect test samples. In particular, in the experiments, from the camera a point-cloud of the scene is acquired, the plane segmented, and the mask of the DMLO object generated. Therefore, the mask is processed by the proposed pipeline to extract the topological representation. The manipulation action is computed according to Sec. IV, and the robot is commanded to execute it. The DMLO is configured to have at least one *intersection-point*. The goal of the manipulation task is to remove the intersection, effectively disentangling the different sections of the DMLO. After the manipulation action is executed, the mask is processed again by the proposed pipeline to extract the obtained topological representation. The experiment is repeated 10 times for each of the three types of DMLOs, i.e., *WH1*, *WH2*, and *WH3*. In total, a set of 60 test samples is collected, 20 for each type of DMLO.

The sequence of manipulation steps for each type of the DMLO is displayed in Fig. 11. The figure contains snapshots extracted from a video available as supplementary material. The experiments show that the proposed pipeline is able to correctly identify all the *branch-points* and *sections* of the DMLOs which allowed a correct manipulation action to be computed. The results are also confirmed by the evaluation of the test samples, which is reported in Sec. V-D.

TABLE I

OFFLINE EVALUATION RESULTS IN TERMS OF DICE SCORE FOR THE LINK PREDICTED IN GRAPH $\mathcal{G}''$ AND DETECTION ACCURACY FOR $\mathcal{B}$ AND $\mathcal{I}$

category	link pred dice score	*branch-points* accuracy	*branch-points* (ratio)	*intersection-points* accuracy	*intersection-points* (ratio)
WH1	0.952	1.0	(20/20)	1.0	(10/10)
WH2	0.938	0.952	(60/63)	0.938	(15/16)
WH3	0.944	0.846	(93/106)	0.923	(12/13)
total	0.944	0.915	(173/189)	0.949	(37/39)

D. Offline Evaluation

The offline evaluation is performed on the 60 samples collected in the manipulation experiment of Sec. V-C. The evaluation is performed by comparing the extracted topology representation with the ground truth one. The ground truth is manually annotated by an expert operator. The annotation process consists of manually providing *branch-points*, *intersection-points*, and *sections* labels.

1) Overall Results: The results of the offline evaluation are presented in Tab. I, with values reported for each test class and also as the average across the entire test dataset. In Fig. 12, the input masks used in the experiment trials shown in Fig. 11 are displayed. Notably, the proposed method extracts the DMLO topologies using only those input masks.

When it comes to predicting the edges for $\mathcal{G}''$, the results are expressed through the *dice score*, emphasizing the effectiveness of the approach in generating reliable edges to construct the topology graph.

The proposed pipeline successfully identifies the majority of *branch-points* within the DMLO samples. Referring to Tab. I, it can be observed that for *WH2* and *WH3* samples, there are 3 and 7 instances of false positive *branch-points* detected, respectively. Additionally, the *WH3* sample exhibits 6 false negative *branch-points* detections.

As for the *intersection-points*, there are inaccuracies in the *WH2* sample, with one *intersection-point* incorrectly identified as a *branch-point*, and in the *WH3* sample, where one *intersection-point* remains undetected due to graph-related inaccuracies.

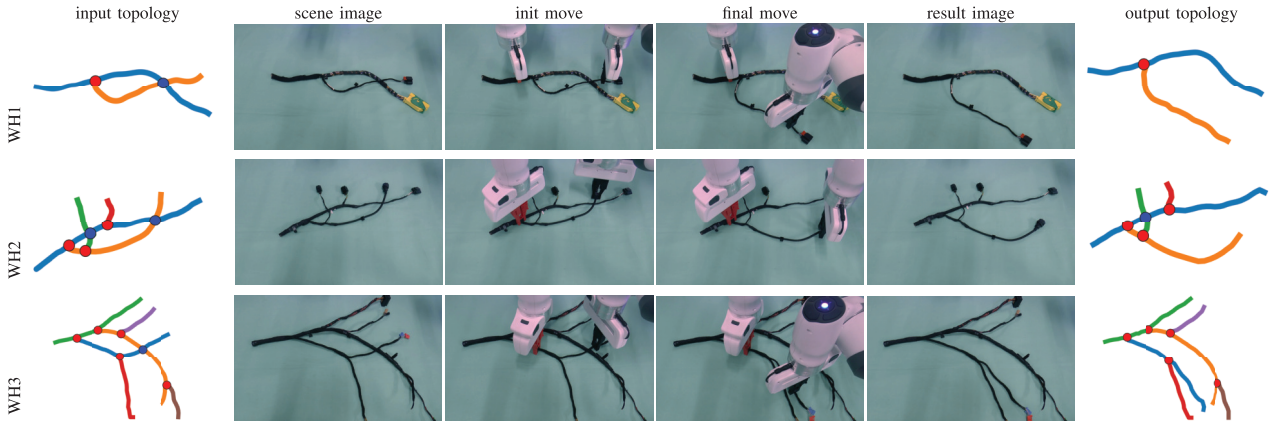


Fig. 11. Snapshots of the untangling manipulation experiment employing three different automotive DMLOs. The *branch-points* and *intersection-points* are shown as red and blue dots.

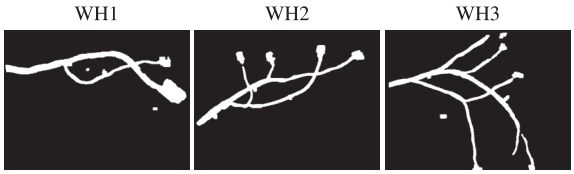
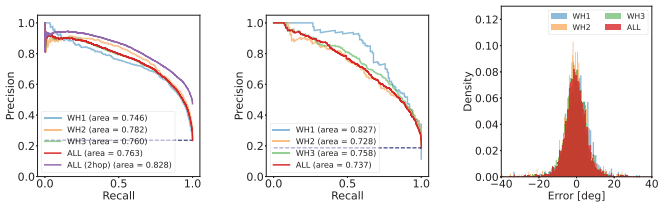


Fig. 12. Example input masks used in experiments.



(a) *link prediction* (b) *node classification* (c) *node orientation*

Fig. 13. Offline evaluation results of the single learned components of the proposed pipeline for the different DMLOs composing the test set. With *ALL*, the entire set of samples is considered.

2) Performance of the Single Components: To provide a more detailed analysis of the performance of each individual learned component within the approach, here are presented separate evaluations for the link predictor (Sec. III-B3), node orientation predictor (Sec. III-B4), and node classifier (Sec. III-B5). The results are shown in Fig. 13. Since all three components are engaged in binary classification tasks, a common evaluation metric, specifically the *precision-recall* curve, is employed.

In the evaluation of the link predictor, ground truth edges from the reference graph are extracted and utilized for assessment. For the node orientation predictor, ground truth node orientations are derived by fitting spline curves along the labeled sections and by determining the tangent lines to the curves at each node's position.

The node classifier, on the other hand, is assessed by utilizing the labeled *branch-points* and *intersection-points*. It is worth noting that the location of the assigned high-degree node point may vary in relation to the node's position in the graph. Consequently, the classifier is expected to provide a one-class prediction within a specified radius, set at two times the thickness of the DMLO in the relevant area.

When examining the outcomes of the link prediction task (as shown in Fig. 13a), it is notable that the predictor faces the challenge of accurately classifying only the two closest neighboring nodes to the considered one. Rather than delivering a sharp distinction, the predictor often produces a more gradual prediction, occasionally assigning high probabilities to the two more distant nodes as well. Indeed, this is clearly illustrated by the *ALL* and *ALL (2hop)* curves in the figure. With *ALL*, an edge is considered positive if connects two nodes at a 1-hop distance along a section of the DMLO. Instead, with *ALL (2hop)* are considered positive edges both the 1-hop and 2-hops ones within the same section of the DMLO. In the proposed methodology, this behavior does not pose an issue, as it effectively filters out unnecessary edges, as elaborated in Sec. III-C1.

Concerning the node classification task, the plot in Fig. 13b provides clear evidence of the capabilities of the classifier in locating the areas of the graph subject to high connectivity.

Conversely, the orientation estimation error, as depicted in Figure 13c, exhibits a distribution akin to the normal curve, with statistical characteristics of approximately 0.63 ± 10.03 degrees. Notably, it is essential to highlight that this error is primarily influenced by incorrect edges within the mask and the presence of obtrusive objects such as clips and plugs.

As a remark, the performance of the learned components (i.e Secs. III-B3, III-B4 and III-B5) are not impacted by variations in the overall number of *branch points* and *intersection points*, as the graph is processed at the local level.

E. Timings and Model Complexity

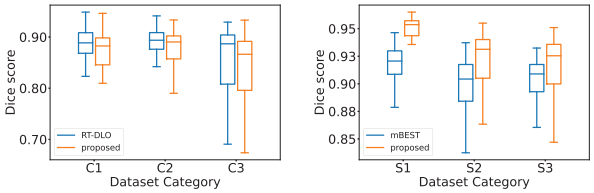
In Tab. II, the recorded timings for the primary procedures of the proposed method across the test set collected in Sec. V-C are presented. The values are expressed in milliseconds and have been obtained on a laptop PC running Ubuntu 20, equipped with an Intel Core i7-12700H processor, 16GB of RAM, and an Nvidia 3050Ti graphics card with 4GB of VRAM, all powered by PyTorch 1.10 with CUDA enabled.

The table provides clear evidence of the efficiency of the approach when executed on a standard hardware configuration. Computational demands increase with scene complexity, as the average graph nodes in *WH1*, *WH2*, and *WH3* are 62, 82, and 140, respectively.

TABLE II

TIMINGS OF THE MAIN PROCEDURES OF THE APPROACH ACROSS THE TEST SET. VALUES IN MILLISECONDS

Procedure	WH1	WH2	WH3	AVG. All
Graph Generation	7.3	8.5	12.1	9.3
Link & Node Orientation Prediction	9.2	9.3	9.6	9.3
Node Classification	9.6	12.3	19.5	13.8
Edges Filtering	4.6	6.0	10.2	6.9
High-degree Node Solver	2.1	3.6	8.1	4.6
Total	32.8	39.7	59.7	43.9



(a) comparison against *RT-DLO* (b) comparison against *mBEST*

Fig. 14. Comparisons of the proposed method against *RT-DLO* [4] and *mBEST* [27] performing the instance segmentation task on the respective proposed DLOs test datasets.

Regarding model complexity, the CNN feature extractor, link predictor, and node orientation predictor have 13K, 21K, and 10K learnable parameters, respectively, while the subgraph node classifier has 11K learnable parameters. Importantly, the graph size does not affect the total number of learnable parameters, as they depend solely on the model architecture.

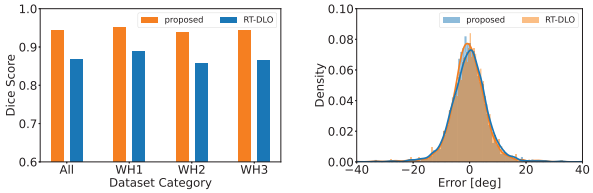
F. Comparison With Methods From DLOs Domain

In the existing literature, there are no direct and comprehensive comparatives available for the proposed pipeline. However, it is possible to benchmark it against methods designed for similar tasks, particularly those related to the perception of DLOs. In this context, two methods emerge as possible benchmarks, namely *RT-DLO* and *mBEST* [27]. Both of these baseline methods focus on instance segmentation of DLOs: starting from a semantic segmentation mask, they produce a colored mask where each DLO is represented by a unique color corresponding to its ID. In particular, *RT-DLO* employs a graph-based representation of the scene similarly to the proposed framework. Instead, *mBEST* exploits a skeleton-based approach.

1) DLOs Instance Segmentation: The proposed method is adjusted to produce a similar output to the one of *RT-DLO* and *mBEST* for what concerns the instance segmentation task. Therefore, given the graph topology, a colored mask is generated based on the extracted *section* with mask thickness information obtained from the distance transform image M_{dist}.

For the evaluation, the same test dataset as the respective works is utilized, which includes dataset categories *C1*, *C2* and *C3* from [4] and *S1*, *S2*, and *S3* from [27]. The evaluation metric employed is the dice score.

The results of these comparisons are shown in Fig. 14 in the form of boxplots. From the plots, it is evident that the proposed method serves as a viable alternative to the baseline methods, with the primary distinction being the versatility of the proposed approach: it can be seamlessly applied in a context involving both DMLOs and DLOs. When compared



(a) link prediction scores (b) node orientation errors

Fig. 15. Ablation study of link prediction and node orientation tasks with components taken from *RT-DLO*.

to *mBEST*, the proposed method exhibits robust performance and the capability to handle self-loops without requiring any modifications compared to the case of DMLOs. Regarding the comparison with *RT-DLO*, similar performance is observed, albeit with a slight decrease in the Dice score values. This drop can primarily be attributed to the differing scopes of application for the two methods: *RT-DLO* focuses on DLOs, while the proposed method targets DMLOs. Specifically, *RT-DLO* only addresses highly-connected nodes as *intersection* points. In contrast, the proposed method accommodates nodes with a degree of three (i.e., *branch* points), which may lead to the estimation of suboptimal topologies when applied to scenes containing DLOs. Therefore, better results could be achieved by tailoring the proposed method specifically for DLOs.

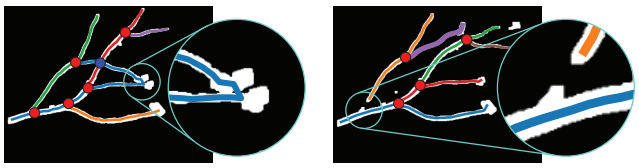
2) Ablation Study: *RT-DLO* approach is similar to the proposed one in terms of graph-based representation. Thus, a study about replacing several components of the proposed pipeline with the alternative ones of *RT-DLO* is conducted here. Specifically, the use of *RT-DLO*'s graph generation and link prediction approach is evaluated to replace the proposed graph initialization stage (Sec. III-A) and the GNN-based link prediction (Sec. III-B3). The results of this evaluation can be seen in Fig. 15a. One notable advantage of the proposed graph initialization stage is its ability to maintain consistent node density, even when dealing with highly diverse masks. In contrast, the vertex sampling ratio is manually adjusted in *RT-DLO* to match the number of nodes in the proposed approach for a fair comparison. Overall, the proposed method demonstrates superior performance in predicting consistent edges within the node-set.

Additionally, the node orientation prediction task in Sec. III-B4 is replaced with *RT-DLO*'s specific neural network approach, which accomplishes the same task but using image crops of M_{dist}. The results are presented in Fig. 15b. The error in node orientation prediction is comparable between the two approaches, with the primary distinction lying in the architecture described in Sec. III-B4, which eliminates the need of mask cropping.

An attempt was also made to utilize the graph generated by *RT-DLO* along with the predicted node orientations to perform node subgraph classification in Sec. III-B5. Nevertheless, the outcomes proved to be highly unsatisfactory, underscoring the crucial importance of the preliminary encoding stage carried out in Secs. III-B3 and III-B4.

G. Limitations and Failure Cases

The proposed approach is not exempt from limitations and drawbacks, which manifest themselves in two primary areas: 1) the representation of topology and 2) the manipulation



(a) merging of different *sections* (b) missing *branch-point*

Fig. 16. Limitations of the proposed method concerning topologies derived from masks that exhibit artifacts and gaps.

strategy. As for the former, key limitations are depicted in Fig. 16. Specifically, Fig. 16a illustrates a scenario where two distinct branch *sections* are merged into a single one, while Fig. 16b illustrates a situation where a *branch-point* is not detected. These issues arise because the DMLO's topology is reconstructed based solely on the input mask, making it susceptible to inaccuracies and suboptimal topologies due to gaps and other mask artifacts. While the graph-based approach does provide a degree of resilience against these issues, it is limited in its effectiveness up to a certain extent. Addressing the first limitation might be achievable by imposing a consistency constraint on the smoothness of the extracted *sections*. Nevertheless, this solution introduces a new parameter to tune and does not represent a definitive resolution. In addition, the topology extraction phase described in Sec. III-C3 relies on a fundamental assumption about node degree to distinguish between *branch-points* and *intersection-points*. However, this approach may become inadequate in scenarios where *branch-points* involve a significantly larger number of connecting *sections*. In such cases, more sophisticated methods will be necessary. Extending the approach from the planar to the 3D domain by incorporating 3D node positions as distinguishing features could simplify the disambiguation between *intersection-points* and *branch-points*.

Regarding the manipulation aspect, the principal limitation originates from the simplicity of the proposed manipulation strategy in Sec. IV. Indeed, the method can fail in the case of complex DMLOs, where the *branch-points* are not easily accessible or there is no space to perform the manipulation action. For example, this occurs for WH2 in Fig. 11, where the small length of one *section* prevents disentangling with the current system due to collision issues between the robots. This simplicity of the manipulation approach is intentional, as the paper's objective is to present a methodology to derive a topological description of a DMLO without prior knowledge of its actual structure. Hence, a straightforward manipulation strategy is chosen to highlight the advantages of using this representation. We believe that the challenges inherent in DMLO manipulation—such as multi-step planning, collision avoidance, dynamic replanning, and handling elastic deformations—require a more comprehensive exploration. The proposed topology representation can serve as a key component for effectively addressing these challenges in future research works.

VI. Conclusion

This paper presents a new approach to efficiently and accurately estimate the topology representation of DMLOs using graph neural networks. Learning on graph data has proven to be effective in exploiting the structure of the DMLO graph for node and edge embedding, which has promising implications for DMLO applications in robotic manipulation systems. The offline evaluation of a set of DMLOs demonstrates that this method can estimate topology representation with good accuracy and respectable time efficiency. Moreover, an online experiment using a robotic cell highlights the practical application of this approach for improved manipulation capabilities in real-world scenarios.

In future works, we will explore the integration of knowledge about the topological structure of the DMLO. This will involve investigating methods to match and align the extracted DMLO topology within the actual scene. Furthermore, we plan to develop a more sophisticated dual-arm manipulation strategy.

References

[1] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Sci. Robot.*, vol. 6, no. 54, May 2021.

[2] J. Trommnau, J. Kühnle, J. Siegert, R. Inderka, and T. Bauernhansl, "Overview of the state of the art in the production process of automotive wire harnesses, current research and future trends," *Proc. CIRP*, vol. 81, pp. 387–392, Jan. 2019.

[3] G. E. Navas-Reascos, D. Romero, J. Stahre, and A. Caballero-Ruiz, "Wire harness assembly process supported by collaborative robots: Literature review and call for R&D," *Robotics*, vol. 11, no. 3, p. 65, Jun. 2022.

[4] A. Caporali, K. Galassi, B. L. Žagar, R. Zanella, G. Palli, and A. C. Knoll, "RT-DLO: Real-time deformable linear objects instance segmentation," *IEEE Trans. Ind. Informat.*, vol. 19, no. 11, pp. 11333–11342, 2023.

[5] A. Caporali, M. Pantano, L. Janisch, D. Regulin, G. Palli, and D. Lee, "A weakly supervised semi-automatic image labeling approach for deformable linear objects," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 1013–1020, Feb. 2023.

[6] X. Huang, D. Chen, Y. Guo, X. Jiang, and Y. Liu, "Untangling multiple deformable linear objects in unknown quantities with complex backgrounds," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 1, pp. 671–683, 2023.

[7] M. Yu, K. Lv, H. Zhong, S. Song, and X. Li, "Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 417–436, Feb. 2023.

[8] N. Lv, J. Liu, and Y. Jia, "Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2341–2353, Aug. 2022.

[9] J. Guo, J. Zhang, D. Wu, Y. Gai, and K. Chen, "An algorithm based on bidirectional searching and geometric constrained sampling for automatic manipulation planning in aircraft cable assembly," *J. Manuf. Syst.*, vol. 57, pp. 158–168, Oct. 2020.

[10] M. Wnuk, C. Hinze, M. Zürn, Q. Pan, A. Lechler, and A. Verl, "Tracking branched deformable linear objects with structure preserved registration by branch-wise probability modification," in *Proc. 27th Int. Conf. Mechatronics Mach. Vis. Pract. (M2VIP)*, Nov. 2021, pp. 101–108.

[11] M. Zürn, M. Wnuk, A. Lechler, and A. Verl, "Topology matching of branched deformable linear objects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 7097–7103.

[12] P. Kicki et al., "Tell me, what do you see—Interpretable classification of wiring harness branches with deep neural networks," *Sensors*, vol. 21, no. 13, p. 4327, Jun. 2021.

[13] K. Galassi and G. Palli, "Robotic wires manipulation for switchgear cabling and wiring harness manufacturing," in *Proc. 4th IEEE Int. Conf. Ind. Cyber-Physical Syst. (ICPS)*, May 2021, pp. 531–536.

[14] Y. Yang, J. A. Stork, and T. Stoyanov, "Tracking branched deformable linear objects using particle filtering on depth images," in *Proc. IEEE 20th Int. Conf. Automat. Sci. Eng. (CASE)*, Aug. 2024, pp. 912–919.

[15] H. G. Nguyen, M. Kuhn, and J. Franke, "Manufacturing automation for automotive wiring harnesses," *Proc. CIRP*, vol. 97, pp. 379–384, Jan. 2021.

[16] T. Hermansson, R. Bohlin, J. S. Carlson, and R. Söderberg, "Automatic assembly path planning for wiring harness installations," *J. Manuf. Syst.*, vol. 32, no. 3, pp. 417–422, Jul. 2013.

[17] F. Yumbla, M. Abeyabas, T. Luong, J.-S. Yi, and H. Moon, "Preliminary connector recognition system based on image processing for wire harness assembly tasks," in *Proc. 20th Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2020, pp. 1146–1150.
[18] X. Zhang, Y. Domae, W. Wan, and K. Harada, "Learning efficient policies for picking entangled wire harnesses: An approach to industrial bin picking," *IEEE Robot. Autom. Lett.*, vol. 8, no. 1, pp. 73–80, Jan. 2023.
[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[20] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NeurIPS*, Jan. 2017.
[21] F. Bullo, *Lectures on Network Systems*, 1st ed., Seattle, WA, USA: Kindle Direct Publishing, 2022. [Online]. Available: https://fbullo.github.io/lns
[22] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2008.
[23] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. NeurIPS*, Jan. 2018.
[24] X. Ma, D. Hsu, and W. S. Lee, "Learning latent graph dynamics for visual manipulation of deformable objects," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 8266–8273.
[25] Y. Deng, X. Wang, and L. Chen, "Learning visual-based deformable object rearrangement with local graph neural networks," *Complex Intell. Syst.*, vol. 9, no. 5, pp. 5923–5936, Oct. 2023.
[26] A. Longhini et al., "EDO-Net: Learning elastic properties of deformable objects from graph dynamics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 3875–3881.
[27] A. Choi, D. Tong, B. Park, D. Terzopoulos, J. Joo, and M. K. Jawed, "MBEST: Realtime deformable linear object detection through minimal bending energy skeleton pixel traversals," 2023, *arXiv:2302.09444*.
[28] A. Caporali, K. Galassi, and G. Palli, "Deformable linear objects 3D shape estimation and tracking from multiple 2D views," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3851–3858, Jun. 2023.
[29] K. Lv, M. Yu, Y. Pu, X. Jiang, G. Huang, and X. Li, "Learning to estimate 3-D states of deformable linear objects from single-frame occluded point clouds," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 7119–7125.
[30] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.
[31] G. Borgefors, "Distance transformations in digital images," in *Proc. Comput. Vis., Graph., image Process.*, vol. 34, Jun. 1986, pp. 344–371.
[32] C. R. Qi, Y. Li, H. Su, and L. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. NeurIPS*, Jan. 2017.
[33] M. Denninger et al., "BlenderProc," 2019, *arXiv:1911.01911*.
[34] X. Yang and J. Yan, "Arbitrary-oriented object detection with circular smooth label," in *Proc. ECCV*. Cham, Switzerland: Springer, Jan. 2020, pp. 677–694.
[35] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.
[36] J. You, R. Ying, and J. Leskovec, "Design space for graph neural networks," in *Proc. NeurIPS*, Jan. 2020.

Alessio Caporali received the M.Sc. and Ph.D. degrees in automation engineering from the University of Bologna, Italy, in 2019 and 2024, respectively. He is currently a Junior Assistant Professor (fixed-term) with the University of Bologna, working at the intersection of robotics and machine learning. His research interests include computer vision and tactile sensing, deformable object modeling, robotic manipulation of deformable objects, and deep learning for physical systems.

Kevin Galassi is currently pursuing the National Ph.D. degree in artificial intelligence with the Laboratory of Robotics (LAR), University of Bologna. He is the Chief Executive Officer of ROBOSECT, a spin-off of the University of Bologna. His research interests include industrial robotic manipulation of deformable objects, collaborative robotics, and robotic learning.

Riccardo Zanella received the M.Sc. degree (Hons.) in control systems engineering from the University of Padova in 2016 and the Ph.D. degree in biomedical, electrical, and system engineering from the University of Bologna in 2021. He is currently a Post-Doctoral Researcher with the Robotics and Mechatronics (RaM) Group, University of Twente. His research focuses on developing optimal control techniques for robotic systems interacting with unstructured environments. He integrates machine learning with physical knowledge, such as energy flow and geometric structures.

Gianluca Palli (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in automation engineering from the University of Bologna, Bologna, Italy, in 2003 and 2007, respectively. He is currently a Full Professor with the University of Bologna. He has authored or co-authored more than 180 scientific articles presented at conferences or published in journals. His research interests include design and control of manipulation devices, mobile manipulation, manipulation of deformable objects, and soft robotics.