Ariadne+: Deep Learning–Based Augmented Framework for the Instance Segmentation of Wires

Alessio Caporali, Riccardo Zanella, Daniele De Greogrio, and Gianluca Palli, *Senior Member, IEEE*

***Abstract*—In this article, an innovative algorithm for instance segmentation of wires called Ariadne+ is presented. Although vastly present in many manufacturing environments, the perception and manipulation of wires is still an open problem for robotic applications. Wires are deformable linear objects lacking of any specific shape, color, and feature. The proposed approach uses deep learning and standard computer vision techniques aiming at their reliable and time effective instance segmentation. A deep convolutional neural network is employed to generate a binary mask showing where wires are present in the input image, then the graph theory is applied to create the wire paths from the binary mask through an iterative approach that aims to maximize the graph coverage. In addition, the B-Spline model of each instance, useful in manipulation tasks, is provided. The approach has been validated quantitatively and qualitatively using a manually labeled test dataset and by comparing it against the original Ariadne algorithm. The timings performances of the approach have been also analyzed in depth.**

***Index Terms*—Computer vision, deep neural networks, industrial manufacturing, instance segmentation.**

I. Introduction

THE development of robotic solutions for the manipulation of deformable objects is a topic of interest in several domains of industrial manufacturing not only in the automotive [1], [2], aerospace [3], and textile industries [4] but also in other very diverse fields as robotic surgery [5], [6] and food processing [7]. Nowadays, human work is intensively adopted in these domains due to the complexity involved in the manipulation tasks of objects that may have both unpredictable initial configurations and large deformability and plasticity.

Deformable linear objects (DLOs) are a particular subgroup of deformable objects consisting of wires, cables, strings, ropes, and elastic tubes. Although vastly present in every industrial environment, wires and wiring harnesses still represent a problematic task for robotic applications. This is the result of few peculiarities embedded in these objects, like not having any specific shape (due to the deformability), nor color, nor any relevant feature that can make them easily distinguishable with respect to other objects.

This article is motivated by the lack of any reliable and efficient system for the segmentation in instances of wires from complex scenarios, such as in industrial manufacturing. In this article, an algorithm addressing this instance segmentation problem effectively and robustly is presented. The proposed algorithm can be then employed in all those subtasks where an accurate perception of wires is required, such as wire routing, terminals insertion, and cable grasping.

The developed algorithm is addressed as *Ariadne+*. It takes as input data an image of the scene and finds the single instances of each wire in it by combining deep learning and computer vision techniques. Besides, the proposed solution is meant to be time effective. A deep convolutional neural network (DCNN) performs the semantic segmentation of the source image providing as output a binary mask. A computer vision pipeline processes the input image and its binary mask in order to detect the individual instances of the wires by exploiting a superpixel segmentation step to reduce complexity. Then, a region adjacency graph (RAG) is created to manage the superpixel structure and properties efficiently. The RAG is simplified, refined, and clustered, thereafter a procedure is applied to identify the path of each wire. Finally, an assessment of the wires layout is obtained by a task-tailored specific DCNN. The obtained instances are ultimately modeled by B-Splines providing a useful representation for manipulation tasks.

The main contributions of this article can be summarized as follows:

1) A reliable and time effective complete approach for the instance segmentation of wires in real scenarios.
2) Modeling of the detected wires in terms of B-Splines.
3) An open source implementation of the entire algorithm available online [8].

Manuscript received 8 September 2021; revised 5 November 2021 and 21 December 2021; accepted 20 February 2022. Date of publication 25 February 2022; date of current version 30 September 2022. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant 870133 as a part of the RIA Project REMODEL (Robotic tEchnologies for the Manipulation of cOmplex DeformablE Linear objects). Paper no. TII-21-3908. *(Corresponding author: Alessio Caporali.)*

Alessio Caporali, Riccardo Zanella, and Gianluca Palli are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: alessio.caporali2@unibo.it; riccardo.zanella2@unibo.it; gianluca.palli@unibo.it).

Daniele De Greogrio is with the EYECAN.ai, 40122 Bologna, Italy (e-mail: daniele.degregorio@eyecan.ai).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3154477.

Digital Object Identifier 10.1109/TII.2022.3154477

4) A comprehensive experimental validation in terms of segmentation capabilities and timings performances of the approach.
5) A comparison against *Ariadne* [9], the current state-of-the-art segmentation algorithm for DLOs.

II. Related Work

Many research articles on the manipulation of DLOs have been developed, e.g., for knot tying [10]. However, the problem of their robust detection is still underdeveloped [11]. The visual perception of DLOs has been typically addressed in fairly simple settings. In [12], the tracking of DLOs is presented where the images are segmented relying on simple color information. In [13], a good contrast between foreground and background is assumed to be available. Alternatively, in [1], the utilization of augmented reality markers for the detection of wiring harness endpoints is proposed.

Similarly to image processing, also in the RGB-D domain, color information and plane fitting techniques are used to segment the target DLO as in [14] for rope untangling, and [15], concerning the modeling of DLOs via B-Splines chained multiple random models.

Ariadne [9] can be considered the current state-of-the-art method for cables detection and segmentation in images featuring complex backgrounds. It is able to perform both a semantic segmentation and B-Spline modeling for multiple DLOs in the scene. However, it has few drawbacks, as: it requires a neural network (or a manual intervention) to specify the cables start and endpoints; it is intrinsically slow due to the exploration process involved; its performances are heavily affected by perspective settings; and it is not robust to specific background/foreground color combinations. With respect to *Ariadne*, the following improvements are implemented in *Ariadne+*.

1) No manual intervention is needed to initialize the algorithm.
2) Our approach works even in case the endpoints are not present in the image.
3) The superpixel segmentation and graph generation are not performed over the entire image, significantly reducing the execution time.
4) A greater robustness to complex and diverse real scenarios is achieved via novel neural network methods.

To summarize, *Ariadne+* improves *Ariadne* in terms of applicability, accuracy, and execution time.

Besides *Ariadne* [9], the semantic segmentation of DLOs (in particular, wires and cables) has been recently addressed via learning-based methods in [16] where a dataset generation approach is presented and the built dataset made publicly available. In the context of learning-based instance segmentation of objects [17], [18], the major problem in their application to DLOs resides in the lack of publicly available datasets, and consequently, the difficulty in annotating a large set of images. Some approaches are emerged focusing on synthetic data generation pipelines [19], [20] but their applicability to DLOs still needs to be proven due to the existing sim-to-real gap [16].

Although for cables, and DLOs in general, the literature concerning perception algorithms is quite reduced, this is not the case for other domains where the treated objects may have some similarities with cables, such as vessel [21] and suture threads [22] segmentation in medical images. In fact, DLOs appear as tubular structure in images. Hence, algorithms developed for curvilinear structures can find, in theory, application to cables. Regarding the medical domain, in [21], the authors propose a new curvilinear structure segmentation network and show their application to vessel segmentation task. In [22], an approach consisting on the identification of the suture thread tips and a novel "marching" algorithm is presented aiming at segmenting the thread from the background.

Among simpler general "filters" for tubular structures, the most popular one, commonly addressed as *Frangi* filter, is [23], that is a multiscale procedure able to highlight tubular structures. In addition, a well-known method is the *Ridge* filter [24], which is an algorithm used to extract image ridges, commonly applied to medical images showing vessel structures. Both the *Frangi* and *Ridge* filter are based on the Hessian matrix, hence, many false positives are detected in case of complex backgrounds [9]. Finally, ellipse and line segment detector (ELSD) [25] is an algorithm developed for detecting line segments and elliptical arcs. Also ELSD suffers in case of complex backgrounds [9].

III. *Ariadne+* Algorithm

The main steps of the *Ariadne+* algorithm can be summarized as follows.

1) *Semantic segmentation*: A DCNN segments the input image and provides a binary mask M_b as output.
2) *Superpixels segmentation*: The input image is segmented into superpixels taking advantage of M_b.
3) *Graph generation*: An RAG is created to manage superpixels efficiently in terms of neighborhood search and endpoints identifications.
4) *Graph simplification*: The graph's nodes degree is used to find intersection nodes, simplifying them to a single equivalent node.
5) *Graph clustering*: The graph is divided into clusters based on a connectivity measure.
6) *Intersection score evaluation*: In case intersection nodes are present, the scores of their neighbors couples are evaluated using a DCNN called *TripletNet*.
7) *Paths finder*: A paths finding strategy is executed to discover the path of each wire and maximize the coverage of the graph.
8) *Paths layout inference*: A DCNN is used to predict the wires layout in the intersection areas to create correct instance masks.
9) *B-Spline modeling*: The computed paths are approximated by B-Spline curves.

Fig. 1 provides an overview of the algorithm pipeline. In the following, each of those steps is discussed in details.

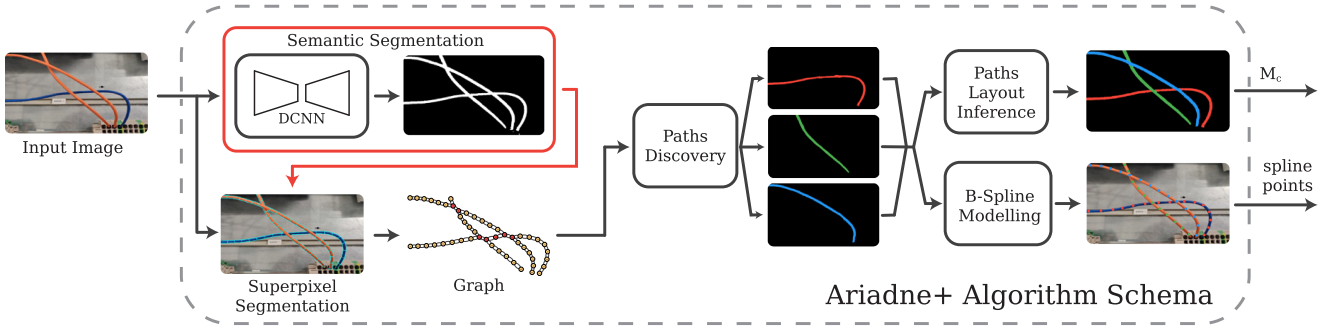


Fig. 1. *Ariadne+* pipeline.

A. Semantic Segmentation

As starting point of the proposed algorithm, a DCNN performs the semantic segmentation of the input image, producing a binary mask M_b of the wires. The DeeplabV3+ neural network model [26] is chosen for that purpose since it represents the state-of-the-art in semantic segmentation. However, it is worth remarking that any model capable of extracting a good quality mask from the scene can be employed in the *Ariadne+* pipeline since no specific constraints are introduced in this regard. As dataset, the *Electric Wires Image Segmentation* dataset [27] is employed. It consists of around 30 K samples of synthetic images of wires of different shapes and colors obtained using a novel approach based on a Chroma-Key method with background swapping [16].

B. Superpixels Segmentation

The superpixel segmentation of images consists in partitioning them into local meaningful subregions by capturing local similarity among the pixels. The aim of this procedure is to simplify the image and speeding processing up. There exist two main branches of superpixel segmentation algorithms: graph-based methods [28] and gradient-ascent approaches [29]. In the proposed algorithm, the simple linear iterative clustering (SLIC) segmentation algorithm [30] is adopted. This solution generates superpixels by clustering pixels based on their color and proximity in the image plane using a 5-D space. Since a binary mask discerning the wires present in the image is available, the algorithm called MaskSlic [31] derived as an extension of the original SLIC and able to perform the clustering only within regions of interest is exploited for our purposes. Thus, the region of interest I'_S, i.e., the wires, on the input image I_S is partitioned into subregions denoted by R_i, i.e., the superpixels, such that $I'_S = \cup R_i$, where $i = 1 \dots r$. From the computed superpixel labels, the centroid information is extracted from each superpixel. The superpixel centroids are then used in the last pipeline step as reference points for wires' B-Spline interpolation. Fig. 2 displays the result of the superpixelization on the sample image used through this article.

C. Graph Generation

In this step, an undirected and not weighted RAG is built from the image superpixel segmentation, where each superpixel becomes a graph node. Hence, the graph is composed of r nodes, where r is the original number of superpixels. The undirected unweighted RAG is denoted by $G = (V, E)$, where V is the set of nodes (or vertices) v_i, corresponding to each region R_i, and E is the set of edges $e_{j,k}$ such as that $e_{j,k} \in E$ if R_j and R_k are adjacent. We also denote as adj(v_i) the set of nodes adjacent to v_i. The RAG generation and its usage should be as most efficient as possible to make the pipeline's run-time execution faster. The edges of the graph are, instead, computed exploiting the neighbors of each superpixel.

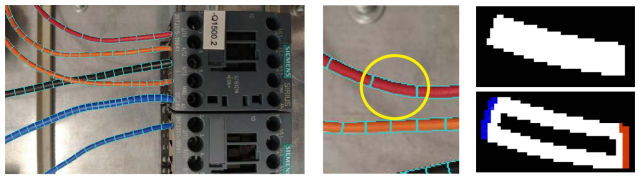


Fig. 2. Superpixelization. (a) Superpixelization of the input image shown in Fig. 1. (b) Crop of image (a). (c) Graph edges computation in superpixel mask (top) before and (bottom) after gradient operation. In (c-bottom) are shown the pixels of the neighboring superpixels in red and blue.

Given a node of the graph, its superpixel label is retrieved and a binary mask of the superpixel is generated. A morphology gradient operation is performed on the mask to fetch pixels belonging to the neighboring superpixels. Consequently, from the labels of these neighbors, the knowledge of the neighboring relationship of the considered superpixels is obtained. Thus, an edge connection is established between the node considered and each associated neighboring node. Fig. 2(c) shows an outline of the approach on a sample superpixel.

In the following, the degree of a node, denoted by $d(v_i)$, represents the number of its neighbors, and the following classification is adopted for the nodes on the basis of their degree.

1) v_i is an outlier if $d(v_i) = 0$, and thus, it is removed from the RAG.
2) v_i is an endpoint if $d(v_i) = 1$.
3) v_i is a segment if $d(v_i) = 2$.
4) v_i is an intersection if $d(v_i) > 2$.

Each node is also augmented with the following *attributes*.

1) *Label ID:* To link each node to the original superpixel.
2) *Centroid coordinates:* Defining the centroid point of the corresponding superpixel in the image.

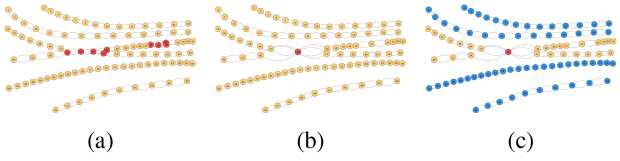


Fig. 3. (a) Graph with intersection nodes highlighted in red. (b) Graph simplified. (c) Graph divided into intersection-free clusters (blue) and cluster with intersection.

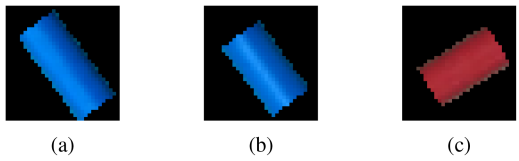


Fig. 4. Triplet loss is a distance-based loss that operates on three inputs. (a) Anchor data. (b) Positive data example (similar to the anchor). (c) Negative data example.

3) *Intersection:* 0 (zero) if node is not an intersection (degree < 2), 1 (one) otherwise.

D. Graph Simplification

Given the RAG initialized in the previous stage from the point of view of nodes and edges, the simplification of the graph is carried out in this pipeline step by exploiting the degree property of the nodes. An *intersection* is defined as an area of the graph composed by one or more intersection nodes. Exploiting the RAG, the intersections nodes are easily identified and organized, based on neighboring relationships, into groups (*intersections*) characterizing a given area of the image.

The graph is then simplified by replacing each intersection group with a single equivalent node that shares the same neighboring relationships as the original group, e.g., the one of all the nodes combined. This simplification makes the path discovery on the graph simpler, in particular, in case of large intersections. Fig. 3(b) shows the result of the simplification applied to the initial situation depicted in Fig. 3(a).

E. Graph Clustering

At this stage, RAG nodes appearing to be closer to each other on the basis of some similarity measure are grouped together. The similarity measure is usually a topological criteria, e.g., the graph structure. In the proposed method, the graph is clustered based on connected components, i.e., each cluster is connected if a path from any point to any other point of that given cluster exists. The result of the clustering is to split G into clusters (subsets) $\mathcal{C}_i$, $i = 1, \ldots, c$, where c is the number of clusters, such that $G = \cup \mathcal{C}_i$ that can be classified into the following two main groups:

1) *intersection-free* clusters, i.e., not having any intersection node inside, shown in blue in Fig. 3(c);
2) clusters with *intersections*, shown in yellow in Fig. 3(c) with the intersection node marked in red.

Each cluster will be then processed as described in Section III-G.

F. Intersection Score Evaluation

The evaluation of the intersections is performed through a partially learning-based predictor. To this end, the intersection nodes are collected in the set V_{int} and the nodes adjacent to an intersection are grouped into a set called $\mathcal{N} := \{v \in \mathcal{C} : d(v) \leq 2 \wedge \text{adj}(v) \cap V_{\text{int}} \neq \emptyset\}$, with $n_i \in \mathcal{N}$, $i = 1 \dot{s} m$, being m the number of elements in $\mathcal{N}$ (i.e., $m = \#\mathcal{N}$). Thereafter, a square matrix $\mathcal{W}_{\text{pred}} \in \mathbb{R}^{m \times m}$ is built and initialized to zero. In case no intersection nodes are present in the RAG, this procedure ends immediately and an empty $\mathcal{W}_{\text{pred}}$ is produced as output. Otherwise, $\mathcal{W}_{\text{pred}}$ will contain the *predictions* among each couple of neighbors of each intersection node $v_{\text{int}_j} \in V_{\text{int}}$, where the generic element w_{kl} of $\mathcal{W}_{\text{pred}}$ describes the value obtained for the pair of nodes $\{n_k, n_l\}$, where $n_k, n_l \in \mathcal{N}$. It follows that the matrix $\mathcal{W}_{\text{pred}}$ will have a strictly positive value (< 1 as described in the following) only if n_k and n_l are neighbors of the same intersection point v_{int_j}.

The values of $\mathcal{W}_{\text{pred}}$ are calculated employing a data-driven procedure, that is preferred with respect to hand-tuned criteria. For this reason, a DCNN called *TripletNet* is developed to perform this prediction. The name of this DCNN is inspired by the fact that the triplet loss criterion [32] is exploited for the required optimization. An example of the input samples of the network is provided in Fig. 4.

The structure of *TripletNet* is composed by a feature extractor (*ResNet18*) and a fully connected layer ($FC_{512,256}$) outputting the embedding representation. The 18-layer version of *ResNet* [33] is selected as a good tradeoff between model complexity and quality of the result, a detailed comparison between the models is reported in Section IV-A. At inference time, the patches x^{n_k} and x^{n_l} are obtained from input nodes n_k and n_l, respectively, by performing two crops of size 32×32 in the source image around each node centroid.

TripletNet computes the distance between each pair of patches, as $d = \|f(x^n) - f(x^{n_l})\|_2^2$. Then, a sigmoid activation function is used to translate the distance into a probability-like value constrained between 0 and 1, where probability 1 is associated to zero distance. Thus, s_{kl}^t denoting the score associated to the pairs of nodes n_k and n_l is obtained. In this way, we assign a score to each prediction coming from the network: in case of very similar patches, their associated score is close to 1, otherwise it is near to 0.

In addition to the prediction performed by *TripletNet*, we calculate the curvature between n_k and n_l, similarly to what shown in [9], and we assign a score s_{kl}^c based on this curvature calculation. In fact, a small ordered sequence of nodes is built by looking at the single neighbors of n_k and n_l, here, denoted as n_k^n and n_l^n. The constructed sequence is, thus, $\mathcal{P}_j = \{n_k^n, n_k, n_l, n_l^n\}$. If the full sequence cannot be built (i.e., is not possible to retrieve n_k^n, n_l^n, or both), a shorter version devoted of one or both of them can be adopted. In fact, the only requirement for the sequence is to contain three nodes. Thus, in extreme cases, although very unlikely, the intersection node can be adopted as replacement, obtaining $\mathcal{P}_j = \{n_k, v_{\text{int}_j}, n_l\}$. In general, a sequence without the intersection node is preferred since it better approximates

the curvature of the wire in the region, i.e., it is more robust with respect to spurious location of the nodes. By considering the consecutive edges between each pair of nodes in $\mathcal{P}_j$, we calculate the angles ϕ_r, with $r = 1\dot{s}\#\mathcal{P}_j - 2$. We deploy the Von Mises distribution ($\mathcal{M}(\cdot)$) for converting the angles information into a score value obtained as $s^c_{kl} = \frac{1}{\#\mathcal{P}_j - 2} \prod_r \mathcal{M}(\phi_r - \phi_{r+1})$.

The curvature score is used to penalize the score computed by *TripletNet* by multiplying the latter with the first

$$s_{kl} = s^v_{kl}\, s^c_{kl} \tag{1}$$

with s_{kl} representing the final prediction score associated to nodes n_k and n_l and inserted to the corresponding entry k, l of $\mathcal{W}_{\text{pred}}$.

For a given intersection node v_{int_j}, the prediction is performed considering all the possible h pairs of neighbor nodes, with $h = \binom{t}{z}$, t denoting the number of items in the set, i.e., the number of neighbors of v_{int_j}, and z describing the number of items forming the combinatorial set (i.e., in our case equal to 2). Thus, we compute the prediction score for every kl pair in h and updating $\mathcal{W}_{\text{pred}}$ accordingly. Thereafter, the procedure is repeated for each intersection node in G.

G. Paths Finder

From the theoretical point of view, a path $\mathcal{P}$ over a generic cluster $\mathcal{C}$ is a sequence of distinct alternating nodes and edges $(v_{i_1}, e_{i_{1,2}}, v_{i_2}, e_{i_{2,3}}, \ldots, v_{i_{l-1}}, e_{i_{l-1,l}}, v_{i_l})$, where an edge $e_{i_{j,k}}$ connects nodes v_{i_j} and v_{i_k}. To simplify the notation, we will refer to the ith path as $\mathcal{P}_i = \{v_{i_1}\dot{s}v_{i_l}\}$, where l is the total number of nodes denoting the path i. The goal is to extend the path node by node in such a way that every node introduced in the sequence belongs to the same wire in the input image. Nodes v_{i_1} and v_{i_l} will be denoted as endpoints. It is worth mentioning that, as result of Section III-E, in the reminder of this section, we will focus our discussion to a single cluster of nodes $\mathcal{C}$, which represent a subset of the entire set of nodes of graph G. The procedure explained is then carried out for every cluster in G.

With reference to Algorithm 1, the generic cluster $\mathcal{C}$ is first scanned to find the set of candidate endpoints $\mathcal{E}$, i.e., the set of nodes having $d(\epsilon_i) = 1$, where $\epsilon_i \in \mathcal{E}$ is used to refer to the ith endpoint candidates. In case $\mathcal{C}$ is an intersection-free cluster (see Section III-E), the set of endpoints will have two elements only, and the path discovery is started directly from one of the two cluster endpoints indifferently. Then, the nodes are connected in sequence exploiting neighboring relations until the second endpoint of the cluster is reached, and the resulting path is added to the set of all the complete paths $\mathbb{P}$.

However, in case the cluster $\mathcal{C}$ presents some intersections, the set $\mathcal{N}$ including all the nodes that are neighbors of intersection nodes is considered. Thereafter, the procedure 2 creates a partial path $\mathcal{P}$ starting from an endpoint ϵ and adding neighbor nodes to this path until a node in $\mathcal{N}$ is reached. It is worth mentioning that this is the only option because, if an endpoint node would be reached, this means that this part of the cluster could be classified as intersection free. Then, the last added node is removed from $\mathcal{N}$ and the partial path $\mathcal{P}$ is added to the set of all the partial paths $\mathbb{P}_{\text{tmp}}$. The procedure is then repeated for each endpoint in the cluster. This will cover all the partial paths from the endpoints to an intersection, but will not cover the partial paths between two intersections. For this reason, new partial paths are then created considering as starting nodes the ones remaining in $\mathcal{N}$ and added to the set partial paths $\mathbb{P}_{\text{tmp}}$. Once these steps are concluded, the set of neighbor nodes should be empty, i.e., $\mathcal{N} = \emptyset$, and the set of partial paths should cover the whole cluster but the intersection nodes, i.e., $\mathbb{P}_{\text{tmp}} = \mathcal{C} \setminus V_{\text{int}}$.

In the last phase of the algorithm, the partial paths are joined on the basis of the intersection score evaluation previously performed; see Section III-F. To this end, the nodes pair (n_1, n_2) associated to the maximum value of $\mathcal{W}_{\text{pred}}$ is selected together with the associated intersection node v_{int} and the two partial paths containing n_1 or n_2 as starting or ending node extracted from $\mathbb{P}_{\text{tmp}}$. These two partial paths are then joined via the intersection node v_{int}, then the resulting path is added to $\mathbb{P}$ if the starting and ending nodes are endpoints; otherwise, it is added back to $\mathbb{P}_{\text{tmp}}$. Then, the rows and columns of $\mathcal{W}_{\text{pred}}$ associated to both the couple (n_1, n_2) and (n_2, n_1) are zeroed to remove them from the selection, and the procedure is repeated until nonzero values are present in $\mathcal{W}_{\text{pred}}$.

Algorithm 1: Paths Finder.

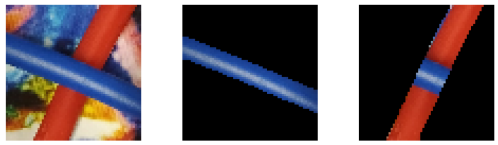
Input: $\mathcal{C}, \mathcal{W}_{\text{pred}}$
Output: $\mathbb{P}$
$\mathcal{E} \leftarrow \{v \in \mathcal{C} : d(v) = 1\}$
$\mathcal{E}_{\text{tmp}} \leftarrow \mathcal{E}$
$V_{\text{int}} \leftarrow \{v \in \mathcal{C} : d(v) > 2\}$
$\mathcal{N} \leftarrow \{v \in \mathcal{C} : d(v) \leq 2 \wedge \text{adj}(v) \cap V_{\text{int}} \neq \emptyset\}$
$\mathcal{N}_{\text{tmp}} \leftarrow \mathcal{N}$
$\mathbb{P} \leftarrow \emptyset$
$\mathbb{P}_{\text{tmp}} \leftarrow \emptyset$
PartialPath($\mathcal{E}, \mathcal{N}, \mathbb{P}, \mathbb{P}_{tmp}$)
$\mathcal{E} \leftarrow \mathcal{E}_{\text{tmp}}$
$\mathcal{N} \leftarrow \mathcal{N}_{\text{tmp}}$
while $\mathcal{W}_{pred} \neq \emptyset \wedge \mathcal{W}_{pred} \neq 0^{k \times k}$ **do**
  $\{n_i, n_j\} \leftarrow \{n_i, n_j \in \mathcal{N} : \arg\max_{i,j} w_{i,j} \in \mathcal{W}_{\text{pred}}\}$
  $v \leftarrow \{v \in V_{\text{int}} : \{n_i, n_j\} \in \text{adj}(v)\}$
  $V_{\text{int}} \leftarrow V_{\text{int}} \setminus v$
  $\mathcal{P} \leftarrow \{\mathcal{P}_i \in \mathbb{P}_{\text{tmp}} : n_i \in \mathcal{P}_i\}$
  $\mathbb{P}_{\text{tmp}} \leftarrow \mathbb{P}_{\text{tmp}} \setminus \mathcal{P}_i$
  $\mathcal{P} \leftarrow \mathcal{P} \cup v$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathcal{P}_j \in \mathbb{P}_{\text{tmp}} : n_j \in \mathcal{P}_j\}$
  $\mathbb{P}_{\text{tmp}} \leftarrow \mathbb{P}_{\text{tmp}} \setminus \mathcal{P}_j$
  if $\#(\mathcal{P} \cap \mathcal{E}) = 2$ **then**
    $\mathbb{P} \leftarrow \mathbb{P} \cup \mathcal{P}$
  else
    $\mathbb{P}_{\text{tmp}} \leftarrow \mathbb{P}_{\text{tmp}} \cup \mathcal{P}$
  for $l \in \{1, \ldots, k\}$ **do**
    $\mathcal{W}_{\text{pred}}[i, l] = 0$
    $\mathcal{W}_{\text{pred}}[l, i] = 0$
    $\mathcal{W}_{\text{pred}}[j, l] = 0$
    $\mathcal{W}_{\text{pred}}[l, j] = 0$
return $\mathbb{P}$

Procedure PartialPath($\mathcal{S}, \mathcal{N}, \mathbb{P}_1, \mathbb{P}_2$).
Input: $\mathcal{S}, \mathcal{N}, \mathbb{P}_1, \mathbb{P}_2$
Output: $\mathcal{S}, \mathcal{N}, \mathbb{P}_1, \mathbb{P}_2$
while $\mathcal{S} \neq \emptyset$ **do**
  $s \leftarrow \mathcal{S}$
  $\mathcal{P} \leftarrow s$
  $\mathcal{S} \leftarrow \mathcal{S} \setminus s$
  $v \leftarrow \text{adj}(s)$
  while $v \notin \mathcal{S} \vee v \notin \mathcal{N}$ **do**
    $\mathcal{P} \leftarrow \mathcal{P} \cup v$
    $v \leftarrow \text{adj}(v) \setminus \mathcal{P}$
  if $v \in \mathcal{S}$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \setminus v$
    $\mathbb{P}_1 \leftarrow \mathbb{P}_1 \cup \mathcal{P}$
  else
    $\mathcal{N} \leftarrow \mathcal{N} \setminus v$
    $\mathbb{P}_2 \leftarrow \mathbb{P}_2 \cup \mathcal{P}$
while $\mathcal{N} \neq \emptyset$ **do**
  $s \leftarrow \mathcal{N}$
  $\mathcal{P} \leftarrow s$
  $\mathcal{N} \leftarrow \mathcal{N} \setminus s$
  $v \leftarrow \text{adj}(s)$
  while $v \notin \mathcal{N}$ **do**
    $\mathcal{P} \leftarrow \mathcal{P} \cup v$
    $v \leftarrow \text{adj}(v) \setminus \mathcal{P}$
  $\mathcal{N} \leftarrow \mathcal{N} \setminus v$
  $\mathbb{P}_2 \leftarrow \mathbb{P}_2 \cup \mathcal{P}$
return $\mathcal{S}, \mathcal{N}, \mathbb{P}_1, \mathbb{P}_2$

H. Paths Layout Inference

The set of paths obtained as solution from Section III-G is not sufficient for performing a correct instance segmentation in case of intersections. In fact, it is not possible to draw the boundary of each wire's instance in the image since each intersection node is assigned to all the paths involved. The goal of this section is to present the deep learning-based approach that handles this last issue. As source of information we use image patches with size 64×64 obtained by cropping the source image around the centroid points associated to the intersection nodes, and then, by postprocessing further the crops. Let us focus our analysis on a sample intersection, as the one depicted in Fig. 5(a). In addition, let us assume that the intersection is constituted by an intersection node shared between two different paths. For each path, a 2-D spline curve is interpolated using the nodes centroid coordinates as control points, and given the spline, a binary mask of the wire corresponding to the path is generated. The crop shown in Fig. 5(a) is, thus, processed utilizing the computed masks obtaining Fig. 5(b) and (c), where the spline-based masks are used to discard all the pixels of the crop not belonging to the considered wire.

The obtained patches are provided as input to *CrossNet*, a deep neural network employed for their classification. It is composed by a feature extractor (*ResNet18*) combined to a fully connected layer ($FC_{512,1}$). Thus, the network performs a binary classification task between two classes (e.g., *is_above* and *is_not_above*) and provides a single probability value as output: 1 (one) if the input patch is predicted to represent a wire placed at the top of the intersection area, 0 (zero) otherwise.



(a) (b) (c)

Fig. 5. (a) Example of intersection node. (b) and (c) Representative patches of the two classes for the displayed intersection. *CrossNet* is employed to predict the class of each patch. (a) intersection. (b) is_above. (c) is_not_above.

Considering again the example shown in Fig. 5, we acquire the predicted probabilities computed by *CrossNet* for both patches of Fig. 5(b) and (c) and select as wire (and path) *is_above* the one with the highest probability. So, the selection is performed based on an output comparison. In fact, since in our framework, there is always one *is_above* class sample among the patches when classifying an intersection, we avoid inserting a threshold for the probability value that can make the approach less reliable (e.g., in case of difficult samples). The approach is not limited to just two wires crossing, but it is applicable also to three or more wires, although less common.

I. B-Spline Modeling

The final paths obtained from Section III-H are employed for estimating B-Splines curves. For every path, the ordered sequence of nodes is translated into a sequence of 2-D points by reading the node's centroid coordinates attributes. The centroid coordinates were computed at the superpixel segmentation stage (see Section III-B). A cubic B-Spline is fitted to these set of points. The obtained curve is thus discretized into an ordered fixed number of 2-D points in pixel coordinates. In this way, we provide a light model of each wire in the image that can be useful, for example, in manipulation and tracking tasks.

IV. Experimental Tests

To validate the proposed algorithm, the original *Ariadne* [9] approach is used as benchmark since, to the best of our knowledge, it is the only tool that allows the instance segmentation of wires from complex images. In addition, this comparison allows us to highlight the improved capabilities of *Ariadne+* over *Ariadne*. The latter requires a CNN for the detection of the wires endpoints. Since we want to make a comparison irrespective of the CNN capabilities, correct endpoints coordinates are directly provided to *Ariadne*.

The rest of this section is divided into an evaluation of the segmentation capabilities (in Section IV-A), a discussion on the superpixels parameters sensitivity (in Section IV-B), an analysis of the timing performances (in Section IV-C), and evidence of the applicability of *Ariadne+* to other types of DLOs (in Section IV-D).

A. Training and Testing

DeepLabV3+ is trained with a ResNet-101 backbone for 200 epochs, with batch size 10, output stride 16, separable convolutions, using Adam for the optimization and employing a polynomial learning rate adjustment policy starting from 10^{-6} to a minimum of 10^{-9}, with power 0.95. The training dataset is obtained from 90% of the electric wires synthetic dataset (see Section III-A), while the validation is done on the remaining 10%. The data augmentation scheme includes hue randomization, channel shuffling, flipping, and finally, resizing (360×640). The early stopping is configured to end the training process when the validation loss does not decrease for five epochs in a row.

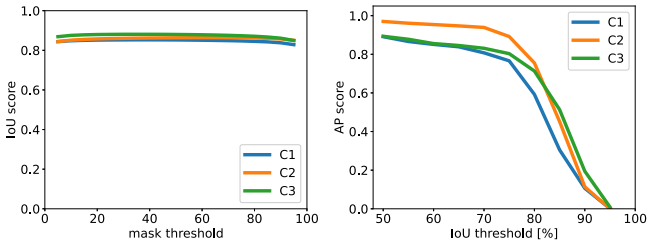
Moreover, *TripletNet*, that is described in Section III-F, is trained for 100 epochs, with batch size 256, using Adam for the optimization and applying a learning rate of 10^{-4}. The dataset used for the training is composed of around 3000 samples organized offline into 4500 different triplets. The usual split of 90-10 for training and validation is used. The data augmentation includes hue, saturation, and value randomization plus channel shuffling. An early stopping strategy is employed by monitoring the validation loss with a patience of ten epochs.

Finally, *CrossNet*, described in Section III-H, is trained in a way similar to the previous network. Hence, for 100 epochs, with batch size 256, using Adam for the optimization and applying a learning rate of 5×10^{-5}. In this case, the dataset is made of around 1500 samples equally divided between the two classes and it is split in the 90-10 way. As data augmentation, alongside hue, saturation, value randomization and channel shuffling, flipping, and random brightness and contrast are employed. Also, during this training, an early stopping strategy is employed with a patience of ten epochs.

The models are implemented in PyTorch 1.4.0 and trained with an NVIDIA GeForce GTX 2080 Ti on an Intel Core i9-9900 K CPU clocked at 3.60 GHz.

To test the proposed algorithm, another dataset of 90 manually labeled images collected in different real scenarios is used. The test dataset is organized into three categories, each containing 30 images.

- *C1* : Scenes with only the target wires laying on a surface and no other disturbing objects. The difficulties in these scenes are the high contrast shadows of the wires, possible chroma similarities between the wires and the background, the dense crosses of wires, the light settings, and the perspective distortions.
- *C2* : Scenes with the target wires on a highly featured and complex background and no other disturbing objects. Here, the challenge for the algorithm is to extract the wires correctly in a cluttered scene.
- *C3* : Scenes with the target wires in a realistic setting as an industrial one (e.g., an electric panel). These can be considered as an example of an application setting, where the difficulties may be given by the metallic surface reflecting the wires and other disturbing objects like commercial electromechanical components characteristic of these panels.



(a) (b)

Fig. 6. (a) IoU computed for the binary mask M_b outputted by the semantic segmentation network when varying the mask threshold ($0.05 : 0.05 : 0.95$). (b) AP computed for M_c when evaluating the IoU $= 0.50 : 0.05 : 0.95$, with M_b thresholded at 0.5.

Each category is further divided into subclasses based on the number of intersections present in the images, i.e., the subcategories 1 (one), 2 (two), and 3 (three) are created.

The algorithm produces a binary mask M_b, which corresponds to the predicted semantic segmentation of the wires, and a colored mask M_c where each instance of the wires is represented by an unique color. We evaluate and compare the outputs by means of the intersection over union (IoU $= \frac{|M \cap M_{gt}|}{|M| + |M_{gt}|}$, where M is the mask we are evaluating and M_{gt} is the ground truth) and average precision (AP) [34] metrics. The latter is computed according to the *primary challenge metric* of [34], i.e., AP at IoU $= 0.50 : 0.05 : 0.95$, with M_b thresholded at 0.5. In Fig. 7 are visible few examples of test images for each category and the output of the proposed algorithm. In Fig. 6(a), the relation between the mask threshold and the IoU score is reported, showing that an almost constant IoU is obtained across all the mask thresholds. This plot justifies our choice of 0.5 as mask threshold for M_b since no major changes are experienced across the different thresholds. In Fig. 6(b), the behavior of the AP score when evaluated at the different IoU threshold values is described.

For the sake of comparison, an intermediate model, named *Hybrid* obtained by combining the path discovery process of *Ariadne* [9] applied on the masked image, i.e. utilizing [26] and [31], is also defined. The *Hybrid* model can be considered to be conceptually halfway between *Ariadne* and *Ariadne+*. In Table I, a comparison between *Ariadne+* and the original *Ariadne* algorithm is performed by analyzing the IoU scores. The values in Table I are obtained by fixing the number of superpixels in *Ariadne+* to 50. Instead, in the case of *Ariadne*, the number of superpixels employed in [30] is set to 750 providing the best results. Table I also reports the comparison between *Ariadne+* and the *Hybrid* configuration through the IoU and AP scores. The number of superpixels is set to 50 also for the *Hybrid* model since it employs the same mask and superpixel segmentation algorithm of *Ariadne+*. The *Hybrid* model requires a set of endpoints coordinates to initialize the search, since the searching algorithm of *Ariadne* is employed. To this end, the same correct endpoints coordinates are provided to the Hybrid model allowing a fair comparison with *Ariadne*. Additionally, the *best case scenario* is analyzed where both the *Hybrid* model and *Ariadne* are provided with the same correct *pairs* of endpoints,

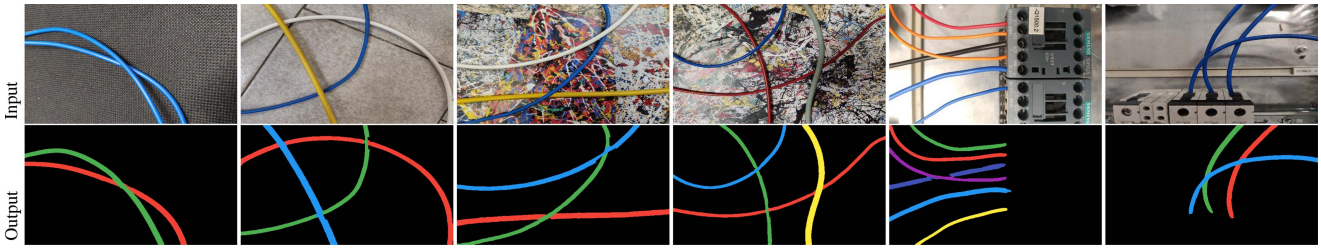


Fig. 7. Qualitative evaluation of the proposed algorithm using two samples for each category (from left two right: *C1*, *C2*, and *C3*).

TABLE I
SCORES ACHIEVED IN EACH TEST CATEGORY (AND SUBCATEGORY) AND THE AVERAGE BY *DEEPLABV3+* (ONLY SEMANTIC SEGMENTATION), *ARIADNE+*, *HYBRID* (COMBINATION OF ARIADNE WITH DEEPLABV3+ AND MASKSLIC), AND *ARIADNE*

Category	#	DeepLabV3+ IoU	Ariadne+ IoU	Ariadne+ AP	Hybrid [9], [31] IoU		Hybrid [9], [31] AP		Ariadne [9] IoU	
	1	85.7	77.8	64.7	64.5	(64.5)	44.5	(48.2)	40.2	(34.9)
C1	2	83.6	77.5	60.8	59.5	(64.6)	36.9	(47.0)	24.5	(33.8)
	3	83.7	74.7	57.3	54.4	(57.7)	26.4	(34.6)	21.0	(33.1)
	1	86.0	82.6	72.6	74.7	(73.4)	59.5	(59.5)	31.0	(22.5)
C2	2	87.3	83.0	71.5	65.6	(66.2)	43.4	(48.0)	17.6	(25.1)
	3	85.0	80.7	66.8	58.6	(60.5)	31.5	(37.8)	12.4	(17.4)
	1	87.9	78.7	70.1	61.2	(65.7)	42.6	(52.3)	30.1	(28.1)
C3	2	88.0	76.8	61.7	62.8	(69.2)	39.6	(51.7)	19.5	(22.8)
	3	85.1	78.8	64.6	61.2	(60.0)	37.2	(40.9)	17.9	(24.5)
average	-	85.8	79.0	65.6	62.5	(64.7)	40.2	(46.7)	23.8	(27.0)

As metrics are used the IoU and AP. The mask M_b computed by *DeepLabV3+* is thresholded at 0.5. The values in brackets denote the *best case scenario*.

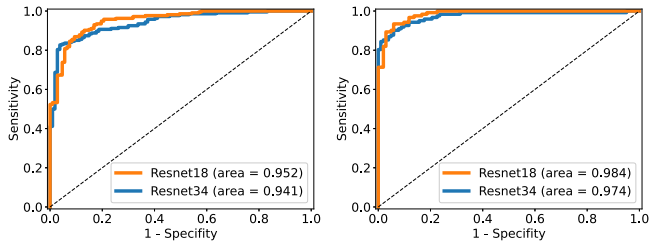


Fig. 8. Receiver operating characteristic (ROC) curves showing the performances of (a) *TripletNet* and (b) *CrossNet* on the respective test datasets using *ResNet18* and *ResNet34* as feature extractors (the other parameters are fixed).

simplifying the searching process. Concerning *Ariadne+*, the knowledge of the endpoints is not required, thus this *best case* analysis is not performed.

Note that for *Ariadne*, the AP scores is not reported since the low values of IoU will result in AP equal to 0 as previously described and reported in [34]. Moreover, for *DeepLabV3+*, the AP score is not usually computed.

From the comparison between *Ariadne+* and *Ariadne*, an average improvement of more than +55 points in the IoU can be noted. The IoU gap between *Ariadne+* and the *Hybrid* model is instead lower, about +16.5 points on average, while the AP score gap is about +25.4 points on average. Thus, the presence of the *Hybrid* models allows us to highlights the following:

1) the importance of the computation of the image mask M_b via [26] combined with [31], which provides a significant boost in the score (70% of the overall improvement in the IoU);
2) the improvement in the novel path discovery algorithm employed in *Ariadne+* compared to the one applied in *Ariadne* and the *Hybrid* model (30% of the gain in the IoU).

The boost due to the segmentation of the background should be expected since a substantial portion of the image is directly avoided in the path discovery, hugely simplifying and speeding up the process. The benefit introduced by the new path discovery algorithm is lower but relevant as absolute value thanks to the presence of the *Hybrid* model. In particular, from the point of view of the number of intersections in the scene, an improvement of +13 and +20 points in the IoU in case of one and three intersections is obtained, respectively. Thus, the proposed path discovery algorithm is more effective with respect to the one adopted in *Ariadne*, especially in the presence of complex scenes with several intersecting cables. Indeed, the consistency of the scores obtained by *Ariadne+* irrespective of the complexity of the scene are reported in Table I, where the standard deviation of the IoU score for *Ariadne+* is 2.56 compared to 5.33 for the *Hybrid* model. Concerning the *best case scenario* previously introduced and denoted with the values in brackets in Table I, although a marginal improvement in case of complex scenes is notable in both *Hybrid* and *Ariadne* model, their performance is still far from *Ariadne+* one. Additionally, the *best case scenario* does not always perform better than the normal baseline, in particular, for *Ariadne*. This is probably due to the limitations of the path discovery module adopted in *Ariadne* when applied on the entire image and on its sensitivity w.r.t. the superpixel parameters.

In addition to the overall algorithm performances, we evaluate the individual performances of *TripletNet* and *CrossNet*. The test datasets for the two networks are derived from the set of 90 test images used to evaluate the overall pipeline. Concerning *TripletNet*, we extract 214 triplets, each consisting of three patches of size 32×32 each, from the intersections available in the test dataset. At test time, for each triplet, we compute the three embeddings, and consequently, the two associated probabilities (anchor-positive, anchor-negative) as explained in Section III-F. To the first probability, a ground truth value of 1 is assigned, while at the second probability, a value of 0 is assigned instead. These assignments are required in order to carry out the AUC–ROC curve analysis. Fig. 8(a) displays the obtained curves comparing the performances with two different feature extractors, highlighting that *ResNet18* is effective for our purposes.

TABLE II
QUANTITATIVE COMPARISON OF SUPERPIXEL PARAMETER SENSITIVITY IN TERMS OF NUMBER OF INTERSECTIONS AND AVERAGE

Spx	#1			#2			#3			average		
	IoU	AP	±	IoU	AP	±	IoU	AP	±	IoU	AP	±
10	61.4	42.6	-26.5	44.6	15.9	-48.7	33.7	5.7	-57.2	46.5	21.4	-44.1
20	72.3	58.8	-10.3	69.1	49.3	-15.3	54.5	28.0	-34.9	65.3	45.4	-20.1
30	77.3	65.7	-3.4	75.5	59.5	-5.1	70.2	49.7	-13.2	74.3	58.3	-7.2
40	79.4	68.7	-0.4	77.2	62.1	-2.5	75.2	57.9	-5.0	77.3	62.9	-2.6
50	**79.7**	**69.1**	**0**	**79.1**	**64.6**	**0**	**78.1**	**62.9**	**0**	**79.0**	**65.5**	**0**
60	79.7	68.9	-0.2	79.8	66.0	+1.4	79.1	64.3	+1.4	79.5	66.4	+0.7
70	79.3	68.3	-0.8	80.6	66.9	+2.3	79.5	65.2	+2.3	79.8	66.8	+1.3
80	78.2	65.9	-3.2	80.2	66.3	+1.7	79.7	65.6	+2.7	79.4	65.9	+0.4
90	76.7	63.0	-6.1	79.3	64.5	-0.1	79.8	65.5	+2.6	78.7	64.3	-1.2

The number of superpixels *Spx* is varied between 10 and 90, whereas the reference value of 50 is shown in bold. As metrics, the IoU and AP scores are employed. With ±, the AP difference compared to the associated reference is highlighted.

Regarding *CrossNet*, a test dataset of 224 samples extracted from the intersections present in the main test set is built. The test set is balanced between the two classes. The samples have a size of 64 × 64 as already described in Section III-H. The choice of this patch size is related to the expected dimensions of the source image (640 × 360), and consequently, to the dimensions of the cables in the scene (i.e., how big the cables are expected to be in the image). The best performances are achieved employing this patch size since it is possible to highlight the intersection area making the network learning process easier. In case of large differences in the images and cables dimensions compared to our settings, the patch size should definitely be modified to accommodate them. Fig. 8(b) shows the AUC–ROC curves obtained for the test set, highlighting that *ResNet18* is the best feature extractor.

Although, in our opinion, the proposed test dataset is statistically relevant for the considered real-world applications, the *Ariadne+* source code is provided together with the aforementioned dataset [8] to allow the testing of the method in scenarios not considered.

B. Superpixels Parameters Sensitivity

As described in Section III-B, *Ariadne+* relays on a superpixel segmentation algorithm to construct the graph representation. Although the employed algorithm to this end [30], [31] exposes several tunable parameters, the only parameter in our pipeline that needs to be adjusted to extract the best result is the number of superpixels in the image. For the experiments shown in this section and reported in Table I, this value is fixed to 50. Instead, for the evaluation reported in Table II, the number of superpixels is varied between 10 and 90 with step of 10 to describe the effects on the overall scores. In particular, the analysis is carried out for each number of intersections highlighting how as the complexity of the scene increases, a higher number of superpixels improves the performances. Table II presents that *Ariadne+* can provide consistent results for a wide range of superpixel number, i.e., from 40 to 90, resulting in a variation of just ±3 points in the AP score. Generally speaking, a low number of superpixels oversimplifies the scene, whereas an unnecessary high value causes the introduction of false intersections. Fig. 9 provides examples for three conditions. It is worth mentioning that the sensitivity of the result to this parameter is somehow mitigated by the graph simplification described in Section III-D.

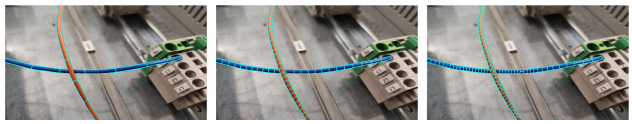


Fig. 9. Qualitative comparison of three different values for the number of superpixels for the same image. From left to right: low (10), normal (50), and high (90).

TABLE III
EXECUTION AVERAGE TIMINGS OF THE DIFFERENT MAIN PARTS OF THE PIPELINE AND TOTAL COMPUTED FOR THE TEST DATASET

Algorithm	mean [s]	standard deviation [s]
Semantic Segmentation	0.018	0.000
Superpixels Segmentation	0.167	0.010
Graph Generation	0.046	0.003
Graph Simplification and Clustering	0.047	0.003
Paths Discovery	0.020	0.002
Paths Layout Inference	0.017	0.002
Total	**0.360**	**0.019**

C. Timings

Table III displays a summary of the average execution timings of the different parts of the pipeline and the total. It is computed for the samples in the test set with the hardware specified in Section IV-A. The superpixel segmentation stage accounts for almost half the total time. On the contrary, the path discovery is very efficient due to the utilization of a batch-inference approach rather than performing the predictions one by one. The same approach is carried out for the paths layout inference, although the amount of gain is less significant due to the lower number of predictions. For Sections III-D and III-E is provided a single time value since the clustering step is implemented very efficiently and its processing time is negligible. Moreover, the average execution time of *Ariadne* [9] is measured on the same samples/hardware as comparison, obtaining an average initialization time of about 1.4 s and a path discovery for each DLO of about 2.5 s. Therefore, it can be concluded that *Ariadne+* speeds up the DLO detection by an order of magnitude.

D. Application to Other Types of DLOs

Ariadne+ is now evaluated with other type of DLOs, as plastic hoses and suture threads, being both commonly found in the medical applications. They share very similar characteristics with cables and wires: they can be approximated with a spline model, they may have intersections, and they cannot have bifurcations. Hence, the applicability of *Ariadne+* also to these types of DLOs is tested without any modification, and Fig. 10 shows the results obtained with plastic hoses and suture threads.

V. LIMIT CASES AND FAILURES

There exist cases where *Ariadne+* is not able to provide a robust and complete solution or a solution at all. Such limit cases arise in particular in the following cases.

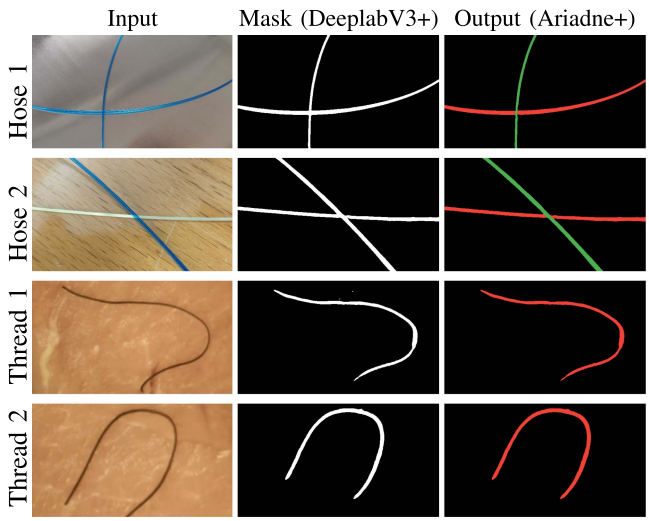


Fig. 10. Examples of *Ariadne+* applied to plastic hoses and suture threads. The latter were provided by [22].

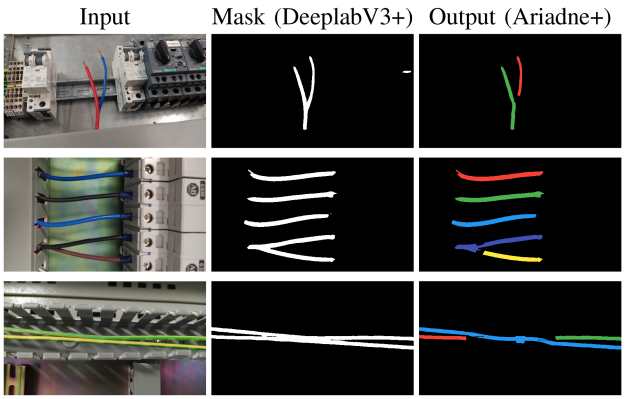


Fig. 11. *Ariadne+:* Limit cases and failures.

1) The aspect ratio between wires thicknesses and image size is much different from the one seen during training. In this case, a fine tuning of the network model may be necessary and the dataset can be obtained via [16].
2) The wires in the image are arranged in specific configurations, as one next to the other without spacing in between. In this case, the algorithm may fail in the creation of a correct graph leading to a wrong result. To solve this problem, we may need to improve the segmentation and superpixelization stages, by optimizing the learning process or by fine tuning the segmentation dataset. An example of this case is shown on the last row of Fig. 11, where the two wires are challenging for the semantic segmentation network.
Ariadne+ tries to solve the task but fails in assessing the correct wires in the scene.

On the contrary, there are cases where *Ariadne+* is able to provide good quality solution. These cases are represented by wires intersecting and merging into a single wire, as shown in the first two rows of Fig. 11. *Ariadne+* solves these images smoothly by first producing candidate paths from the endpoints, and then, by maximizing the allocation of nodes in the most likelihood way.

VI. Conclusion

In this article, we presented a reliable and time effective approach for the instance segmentation of wires. This algorithm may be used in manufacturing environments for all those subtasks involving the perception of wires, as wiring routing, grasping, and terminal insertion. The availability of models of the wires in terms of B-Splines may be useful in those robotic manipulation tasks. The experimental results demonstrated the validity of our method, that is able to robustly provide the instances of the wires even in real complex industrial scenarios. In future works, we will improve the approach by addressing the limit cases explained in Section V and by optimizing the learning process with more data.

References

[1] X. Jiang, K.-m. K. Koo, A. Kikuchi Konno, and M. Uchiyama, "Robotized assembly of a wire harness in a car production line," *Adv. Robot.*, vol. 25, no. 3/4, pp. 473–489, 2011.
[2] 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, 2013.
[3] A. Shah, L. Blumberg, and J. Shah, "Planning for manipulation of interlinked deformable linear objects with applications to aircraft assembly," *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 4, pp. 1823–1838, Oct. 2018.
[4] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Using depth and appearance features for informed robot grasping of highly wrinkled clothes," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 1703–1708.
[5] J. Jayender, R. V. Patel, and S. Nikumb, "Robot-assisted active catheter insertion: Algorithms and experiments," *Int. J. Robot. Res.*, vol. 28, no. 9, pp. 1101–1117, 2009.
[6] J. Pile, G. B. Wanna, and N. Simaan, "Force-based flexible path plans for robotic electrode insertion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 297–303.
[7] R. J. M. Masey, J. O. Gray, T. J. Dodd, and D. G. Caldwell, "Guidelines for the design of low-cost robots for the food industry," *Ind. Robot: An Int. J.*, vol. 37, pp. 509–517, 2010.
[8] [Online]. Available: https://github.com/lar-unibo/ariadne_plus
[9] D. De Gregorio, G. Palli, and L. Di Stefano, "Let's take a walk on superpixels graphs: Deformable linear objects segmentation and model estimation," in *Proc. Asian Con. Comput. Vis.*, 2018, pp. 662–677.
[10] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1141–1150, Dec. 2007.
[11] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey," *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 688–716, 2018.
[12] Y. Wang, D. McConachie, and D. Berenson, "Tracking partially-occluded deformable objects while enforcing geometric constraints," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 14199–14205.
[13] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2372–2379, Apr. 2020.
[14] W. H. Lui and A. Saxena, "Tangled: Learning to untangle ropes with RGB-D perception," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 837–844.
[15] G. Yao, R. Saltus, and A. P. Dani, "Shape estimation for elongated deformable object using B-spline chained multiple random matrices model," *Int. J. Intell. Robot. Appl.*, vol. 4, no. 4, pp. 429–440, 2020.
[16] R. Zanella, A. Caporali, K. Tadaka, D. De Gregorio, and G. Palli, "Auto-generated wires dataset for semantic segmentation with domain-independence," in *Proc. Int. Conf. Comput., Control Robot.*, 2021, pp. 292–298.
[17] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT++ Better real-time instance segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 2, pp. 1108–1121, Feb. 2022.

[18] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "Solov2: Dynamic and fast instance segmentation," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 17721–17732, 2020.
[19] M. Denninger *et al.*, "Blenderproc," 2019, *arXiv:1911.01911*.
[20] W. Qiu and A. Yuille, "UnrealCV: Connecting computer vision to unreal engine," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2016, pp. 909–916.
[21] L. Mou *et al.*, "CS^2-Net: Deep learning segmentation of curvilinear structures in medical imaging," *Med. Image Anal.*, vol. 67, 2021, Art. no. 101874.
[22] B. Lu, X. Yu, J. Lai, K. Huang, K. C. Chan, and H. K. Chu, "A learning approach for suture thread detection with feature enhancement and segmentation for 3-D shape reconstruction," *IEEE Trans. Automat. Sci. Eng.*, vol. 17, no. 2, pp. 858–870, Apr. 2020.
[23] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever, "Multiscale vessel enhancement filtering," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervention*, 1998, pp. 130–137.
[24] J. Staal, M. D. Abràmoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken, "Ridge-based vessel segmentation in color images of the retina," *IEEE Trans. Med. Imag.*, vol. 23, no. 4, pp. 501–509, Apr. 2004.
[25] V. Pătrăucean, P. Gurdjos, and R. G. Von Gioi, "A parameterless line segment and elliptical arc detector with enhanced ellipse fitting," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2012, pp. 572–585.
[26] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.
[27] [Online]. Available: https://www.kaggle.com/zanellar/electric-wires-image-segmentation
[28] P. F. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comp. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.
[29] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2008, pp. 705–718.
[30] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, 2012.
[31] B. Irving, "Maskslic: Regional superpixel generation with application to local pathology characterisation in medical images," 2016, *arXiv:1606.09518*.
[32] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.
[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[34] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

Alessio Caporali received the M.Sc. degree in automation engineering, in 2019, from the University of Bologna, Bologna, Italy, where he is currently working toward the Ph.D. degree in biomedical, electrical and system engineering.

His research interests include computer vision for robotic manipulation of deformable objects.

Riccardo Zanella received the M.Sc. degree in automation engineering from the University of Padova, Padua, Italy, in 2016, and the Ph.D. degree in biomedical, electrical and system engineering from the University of Bologna, Bologna, Italy, in 2021.

He is currently working as a Postdoctoral Researcher with the University of Bologna. His research interests include computer vision and robot learning for autonomous manipulation of deformable objects.

Daniele De Greogrio received the B.Sc. and M.Sc. degrees from the University of L'Aquila, L'Aquila, Italy, in 2008 and 2012, respectively, and the Ph.D. degree in software engineering from the University of Bologna, Bologna, Italy, in 2018.

He was a Postdoctoral Researcher with the University of Bologna in the field of robotic vision. He has been a Software Consultant outside the University for over ten years. He is currently the Co-Founder and the CEO of EYE-CAN.ai with the University of Bologna spin-off company dealing with deep learning and robotics in the industrial field. He is the author of more than 20 publications and 3 patents.

Gianluca Palli (Senior Member, IEEE) received the Laurea 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 coauthored more than 100 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.