This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2023.3245641

RT-DLO: Real-Time Deformable Linear Objects Instance Segmentation

Alessio Caporali, Kevin Galassi, Bare Luka Žagar, Riccardo Zanella, Gianluca Palli and Alois C Knoll

***Abstract*—Deformable Linear Objects (DLOs) such as cables, wires, ropes, and elastic tubes are numerously present both in domestic and industrial environments. Unfortunately, robotic systems handling DLOs are rare and have limited capabilities due to the challenging nature of perceiving them. Hence, we propose a novel approach named *RT-DLO* for real-time instance segmentation of DLOs. First, the DLOs are semantically segmented from the background. Afterward, a novel method to separate the DLO instances is applied. It employs the generation of a graph representation of the scene given the semantic mask where the graph nodes are sampled from the DLOs centerlines whereas the graph edges are selected based on topological reasoning. *RT-DLO* is experimentally evaluated against both DLO-specific and general-purpose instance segmentation deep learning approaches, achieving overall better performances in terms of accuracy and inference time.**

***Index Terms*—Deformable Linear Objects, DLO, Instance Segmentation, Industrial Manufacturing, Computer Vision**

I. Introduction

Deformable Linear Objects (DLOs) belong to the generic class of deformable objects and consist of wires, cables, strings, ropes, and elastic tubes, as the main relevant examples according to [1]. Although vastly present in every domestic and industrial environment, DLOs still represent a problematic task for automated robotic systems, both at perception and manipulation levels [1]. From the perception side, this is a result of the lack of any specific shape, color, texture, or feature making them easily distinguishable with respect to other objects. In addition, DLOs are characterized by small dimensions in terms of diameters, posing an additional challenge concerning their 3 Dimensional (3D) perception capabilities with most sensors [2]. From the manipulation side, the DLOs intrinsic deformability results in a high-dimensional state space with complex and nonlinear dynamics. Thus, modeling and predicting their behavior during a manipulation task is challenging [3], [4].

The problem of DLOs segmentation is usually addressed in simple settings, like color threshold with a single DLO instance [5] or markers [6]. In the last years, several DLO-specific approaches tried to address the instance segmentation problem more steadily, e.g. [7]–[9] with remarkable improvements at the introduction of every novel approach. Although the very recent method named *FASTDLO* [9] reaches good accuracy results with reasonable computation time, i.e. above 20 FPS (Frames Per Second), it is still far from being real-time capable. From the domain of general-purpose Deep Convolutional Neural Networks (DCNN) tackling the instance segmentation task, there exist several approaches real-time capable, e.g. *YOLACT* [10] and *YOLACT++* [11], however applying these methods directly to DLO-like objects usually does not guarantee satisfactory accuracy results [9].

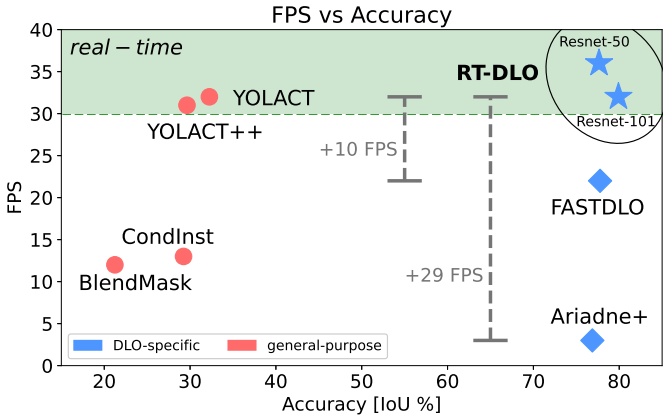


Fig. 1: FPS vs accuracy of *RT-DLO* versus baselines methods.

To mitigate the aforementioned drawbacks and challenges, we propose an algorithm real-time capable and highly accurate for instance segmentation of DLOs, dubbed *RT-DLO* (Real-Time Instance Segmentation of Deformable Linear Objects). In Fig. 1 the plot of FPS vs accuracy shows how *RT-DLO* stands against the competition, being the fastest DLO-specific approach and the most accurate overall on the test-set of [9].

RT-DLO does not require any assumption about the background and the number of DLOs present in the scene. As input, it acquires the RGB image and provides as output a pixel-mapped colored mask where each DLO is represented by a unique color identifying its ID. In addition, being the DLO instances modeled as a sequence of key-points, a representation of the scene with spline curves can be easily obtained, e.g. for manipulation tasks employing a state-space representation different from the image space [3].

First, as a pre-processing step, the input RGB image is propagated through a DCNN trained on synthetically generated data aiming at segmenting the background, i.e. pixels

Alessio Caporali, Kevin Galassi, Riccardo Zanella and Gianluca Palli are with DEI - Department of Electrical, Electronic and Information Engineering, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy.

Bare Luka Žagar and Alois C. Knoll are with the Chair of Robotics, Artificial Intelligence and Real-Time Systems, Technical University of Munich, 85748 München, Germany.

This work was supported by the European Commission's Horizon 2020 Framework Programme with the project REMODEL - Robotic technologies for the manipulation of complex deformable linear objects - under grant agreement No 870133.

Corresponding author: *alessio.caporali2@unibo.it*

not representing a DLO, and providing as output a binary mask. Then, a graph representation of the scene is constructed by efficiently sampling the vertices from the segmentation mask. The edges connecting the graph's vertices are instead computed by reasoning about the topology expressed by the mask, with an approach that considers both the proximity and orientations constraints among the vertices. Ideally, only a maximum of two edges per vertex should be sampled. In the case of intersections of DLOs resulting in the presence of high-degree vertices in the graph, sub-graphs around the target vertices are extracted and further processed to disentangle the DLOs in the graph. Finally, the single DLOs are extracted from the graph based on an analysis of its connectivity. *RT-DLO* achieves a processing rate higher than 30 FPS with an input image of 640×360 pixels. To summarize, the main contributions of this paper are:

1) First instance segmentation approach concerning DLOs able to reach a processing rate higher than 30 FPS, i.e. real-time capable.
2) Robust graph-based enhanced representation of the DLOs configuration in the scene given the segmentation mask.
3) Improved overall performance compared to several baselines, i.e. +2.9% IoU with +7 FPS compared to [9] and +3.4% IoU with +32 FPS compared to [8].

The source code implementing *RT-DLO* and the associated data is available at https://github.com/lar-unibo/RT-DLO.

II. Related Works

A. Real-Time Instance Segmentation

The instance segmentation task consists in predicting objects-wise segmentation masks. Remarkable results in this challenging task were achieved by *Mask R-CNN* [12] with its detect-and-segment approach. However, due to this two-phase method, *Mask R-CNN* is not real-time. Recent approaches for instance segmentation of general objects are [10], [11], [13]–[15]. Among those, only [10], [11], [15] are capable of real-time performances. However, their applicability to DLOs requires attention due to the challenges highlighted in Sec. I. Also considering the dataset supply problem, satisfactory results were obtained only concerning the semantic segmentation task [16] and not for the instance segmentation one. Indeed, the performances of these methods are affected by the DLO instances lacking distinctive embeddings. On the contrary, due to the high-level abstraction by using a graph representation of the DLOs, *RT-DLO* can achieve better performances and robustness.

B. Segmentation of DLOs

The limited adoption of automatic or robotic solutions in the manufacturing and assembly tasks having to deal with DLOs has made the perception of such objects an important research topic of the last decade. In the past, simplifying assumptions were usually made, e.g. knowledge of the background [3], [5], [17], number of DLOs in the scene [5], markers [6].

Specific to DLOs, the first approach tackling complex backgrounds is represented by *Ariadne* [7] which employs a Convolutional Neural Network (CNN) for DLO endpoints detection and a walking algorithm along the superpixels originated from the image. *Ariadne+* [8] improves *Ariadne* in accuracy by employing a DCNN for the background semantic segmentation, removing the need for endpoint detection and thus also significantly speeding up the processing time. In *Ariadne+*, a graph representation of the scene is obtained by exploiting a superpixel-based approach where the graph nodes are selected based on the superpixels centroids and the edges based on superpixels contours overlapping.

Recently, *FASTDLO* [9] was introduced employing a skeleton-based approach on the segmentation mask and a similarity network for the correct interconnection of DLOs segments. *FASTDLO* is currently the state-of-the-art approach for instance segmentation of DLOs, achieving an inference time of more than 20 FPS.

RT-DLO employs an efficient and informative graph representation of the scene as opposed to the skeleton originated *segments*-based approach of *FASTDLO* and superpixel-based one of *Ariadne+*, resulting in faster processing times and improved accuracy, especially at the DLOs intersection. Indeed, *RT-DLO* can handle degraded masks more effectively since the continuity of the segmentation mask foreground along a DLO is not required.

III. Method

The idea exploited in *RT-DLO* is to model the current configuration of the DLOs present in the image with a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and then to extract the DLO instances from the obtained graph. The approach, schematized in Fig. 2, can be subdivided into six main steps:

A) Mask Generation: obtaining a binary mask M_b from the input color image via a DCNN;
B) Vertices Sampling: processing M_b, with vertices orientation characterization employing a CNN;
C) Edges Sampling: exploiting the proximity among the vertices and the orientation between vertices and edges;
D) Intersections Processing: disentangling the DLOs in the graph representation via sub-graphs analysis;
E) DLOs Instances Extraction: computing pixel-wise DLOs instances masks in the image plane;
F) Intersections Layout: assessing the correct instances locally at the intersections.

In the remainder of this section, the procedures for obtaining the graph representation and extracting coherently DLOs instances from it are presented. First, the binary mask M_b generation is discussed in Sec. III-A. Then, concerning the graph formation process, the vertices are examined in Sec. III-B while the edges are in Sec. III-C. Thereafter, the algorithm employed for processing problematic regions of the graph is provided in Sec. III-D. Finally, the extraction of the DLO instances, given the graph representation, is presented in Sec. III-E while the approach for analyzing their layout is in Sec. III-F.

A. Mask Generation

The mask generation step can be considered a preprocessing phase of *RT-DLO* since the graph representation

This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2023.3245641

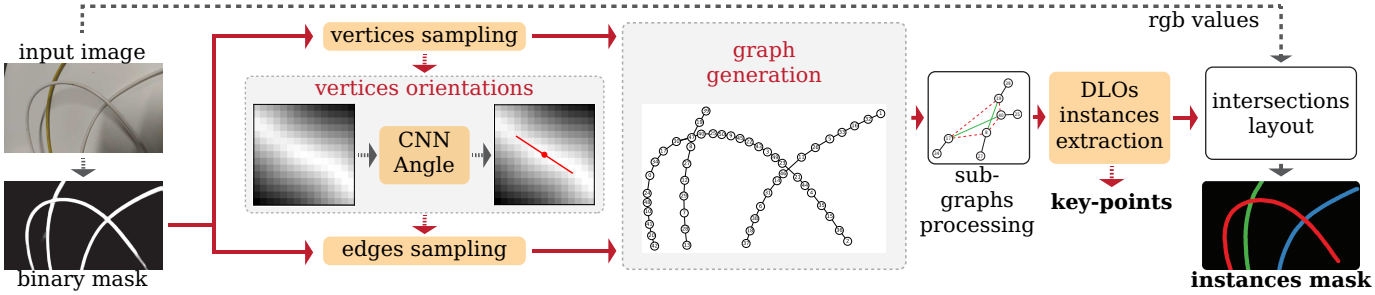


Fig. 2: Schematic representation of the *RT-DLO* algorithm.

of the DLOs is obtained employing only the binary mask M_b of the scene and not the RGB image. In this paper we assume to use a DCNN, specifically DeepLabV3+ [18], trained on synthetically generated data [9]. This choice is convenient since 1) good performances are shown in [9] concerning the semantic segmentation capabilities of this method; 2) a simplification on the comparison of *RT-DLO* against the baseline methods is achieved. Therefore, a binary mask M_b is obtained by setting the pixels predicted to belong to a DLO to 1 and the remaining ones to 0.

It is worth mentioning that *RT-DLO* is independent of the method used to obtain the semantic segmentation mask. Different approaches can be employed depending on application requirements.

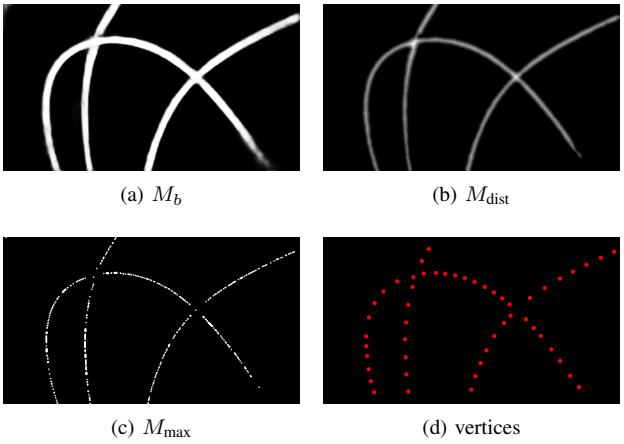
B. Vertices

First, vertices of the graph $\mathcal{G}$ are cleverly sampled from the binary mask M_b and then characterized in terms of local orientation by a CNN.

1) 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. 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) [19], thus assigning an intensity value to each pixel based on the computed distance. In Fig. 3b, M_{dist} originated from M_b (Fig 3a) is shown where M_{dist} is color-mapped on the grayscale level from dark (zero distance) to bright (maximum distance).

Then, M_{dist} is dilated with a small square kernel (i.e. 3×3). The dilation operation is a maximum locating morphology operation. Indeed, as the kernel is convolved over the target image, the maximal pixel value overlapped by the kernel is computed and the corresponding image pixel at the anchor position is replaced. Dilation is usually applied on binary masks to enlarge the foreground (white) portion. Instead, in this paper, the dilation operation is applied to the mask M_{dist} which contains intensities values, i.e. M_{dist} is not binary, obtaining M_{dil}. The local maximums of M_{dist} are retrieved by comparing pixel-wise M_{dist} and M_{dil} masked using M_b, as follows:

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



(a) M_b (b) M_{dist} (c) M_{max} (d) vertices

Fig. 3: Vertices sampling key elements: the mask M_b (a), M_{dist} (b) and M_{max} (c), the obtained vertices (d). The bright regions in (b) denote high intensity values.

Indeed, if the value of pixel (i,j) in M_{dist} and M_{dil} is the same, this means that the considered pixel is a local maximum. By assigning the pixel value of 1 to the maximums and 0 to the rest of the pixels, a new mask is obtained, denoted with M_{max}, and illustrated in Fig 3c. It is worth mentioning that, by construction, M_{max} approximates the center lines of the DLOs in the mask.

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 relatively large and not really tractable in case real-time applications are sought. Thus, the farthest point sampling (fps) algorithm [20] is employed for down-sampling $\mathcal{V}_{\text{max}}$. A sampling ratio of $\alpha \in [0,1]$ is used to specify the amount of down-sampling. The set of vertices $\mathcal{V}$ of the graph $\mathcal{G}$ is obtained as $\alpha\mathcal{V}_{\text{max}}$. In Fig. 3d the vertices extracted from the sample mask of Fig. 3a with $\alpha = 0.15$ are depicted.

2) Vertices Orientations: In the context of linear objects and linear shapes representation, for each given vertex of the graph, an orientation characterization can be performed. The objective is to describe locally the section of the linear object in the vicinity of the vertex as an orientation attribute of the vertex itself. Thus, the local orientation θ of a given vertex at pixel coordinates (x,y) is derived from a local patch of size $\delta \times \delta$ pixels, centered at (x,y) and with intensity values extracted from the distance transform image M_{dist}.

A CNN is used to estimate an angular value from a given patch. Predicting an angular value via a learning-based method

can become quite a complex task due to the periodicity of the angular data resulting in inaccurate distance representations when computing the loss function. Indeed, an angle of 2° describes an orientation quite close both to 5° and 179°, although the corresponding loss values when applying common losses, e.g. L1-loss or MSE-loss, are quite different. An approach pioneered in [21] is thus employed to address the angular periodicity and ambiguity in the loss computation. A given angular value θ in the range $[0°, 180°]$ is encoded as a 180-dimensional vector with entries defined by applying a Gaussian function centered at θ and with variance σ. In this way, the angle θ is propagated smoothly in its proximity enabling benefits during the loss computation. The network structure is composed of two convolutional layers followed by a fully connected linear layer. Each convolution layer comprises a 2D convolution followed by batch normalization. Between the two layers, a max-pooling operation takes place. After the convolution layers, the embedded data are flattened and the fully connected layer is used as an output to classify the patch in the 180-dimensional vector. Binary cross entropy is used as loss function during the training stages, effectively shaping the learning task as a classification problem of the angular value in one of the 180 available classes. Consequently, the actual predicted angle is easily obtained from the 180-dimensional vector as the index of the vector associated to the maximum probability. This angular value characterizes the orientation of the vertex associated to the processed patch.

C. Edges

The set $\mathcal{E} = \{e_j\}_{j=1}^{m}$ contains the m edges of the graph. Identifying the correct edges to be inserted in the graph is a complex task. Indeed, the connections between the vertices should consider both their relative proximity as well as orientation constraints, the latter in the form of vertex orientation and edge orientation. The vertices orientations were described in Sec. III-B2. For convenience, a matrix $E \in \mathbb{R}^{m\times 2}$ describing the edge set $\mathcal{E}$ as organized tuples is introduced.

The relative proximity between vertices is exploited to obtain an initial candidate set of edges, denoted as $\mathcal{E}_{\text{knn}} = \{e_j\}_{j=1}^{m_{\text{knn}}}$. That is, for each vertex, the K_{nn} nearest neighbors in $\mathcal{V}$ are retrieved as edges. The value of K_{nn} is a user-defined parameter and it follows that $m_{\text{knn}} = n \times K_{nn}$ if we consider the edges as directed. In addition, $E_{\text{knn}} \in \mathbb{R}^{m_{\text{knn}}\times 2}$ is the matrix description of $\mathcal{E}_{\text{knn}}$. The K_{nn} nearest neighbor case with $K_{\text{nn}} = 8$ for a sample vertex is depicted in Fig. 4a.

1) Vertex-Vertex Similarity: The orientation constraints between two general vertices v_1 and v_2 are evaluated by assigning a score to their connection by means of the cosine similarity defined as

$$s(\mathbf{d_v^1}, \mathbf{d_v^2}) = \frac{\mathbf{d_v^1}^T \mathbf{d_v^2}}{\|\mathbf{d_v^1}\|\|\mathbf{d_v^2}\|} \tag{1}$$

In particular $\mathbf{d_v^1}$ is obtained as $[\cos(\theta_1), \sin(\theta_1)]^\top$, where θ_1 is the orientation of v_1 obtained from Sec. III-B2. For $\mathbf{d_v^2}$ the derivation is similar. In eq. 1, at the denominator is denoted the product of the norms. The cosine similarity is then used to score the orientations between two vertices pair.

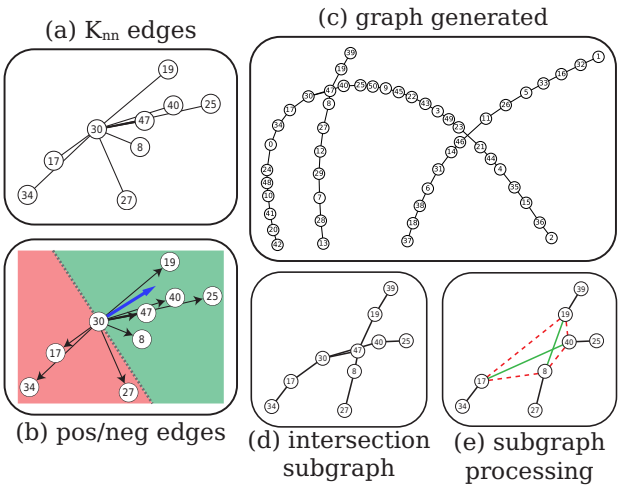


Fig. 4: Edges processing main elements: (a) K_{nn} edges to obtain initial candidate edge set; (b) positive/negative edges illustration; (c) graph generated; (d) intersection subgraph extracted; (e) subgraph processing schema.

For efficiency reasons, the cosine similarity is evaluated by means of matrix operations. Given the matrix $D_v \in \mathbb{R}^{n\times 2}$ of vertices orientations in the form of direction vectors obtained from the predicted angles, i.e. for vertex i we have $\mathbf{d_v^i}/\|\mathbf{d_v^i}\|$, the cosine similarity between each pair of vertices of the set $\mathcal{V}$ can be obtained as

$$S_{v,v} = |D_v D_v^T| \tag{2}$$

being $S_{v,v} \in \mathbb{R}^{n\times n}$ and $|\cdot|$ denoting the absolute value.

2) Vertex-Edge Similarity: Similarly to the vertex-vertex case, the matrix $D_e \in \mathbb{R}^{m_{\text{knn}}\times 2}$ of edges orientations can be defined. It contains the direction vectors obtained by subtracting the coordinates of the associated vertices followed by a normalization by their distance. The cosine similarity between each vertex of $\mathcal{V}$ and each edge of $\mathcal{E}_{\text{knn}}$ is obtained as:

$$S_{v,e} = D_v D_e^T \tag{3}$$

with $S_{v,e} \in \mathbb{R}^{n\times m_{\text{knn}}}$ being the obtained similarity matrix between vertices and edges.

3) Combining $S_{v,v}$ and $S_{v,e}$: At the current stage, because of the dimensions mismatch, it is not possible to combine $S_{v,v}$ (eq. 2) and $S_{v,e}$ (eq. 3). Thus, an augmented similarity vertices score matrix $\bar{S}_{v,v} \in \mathbb{R}^{n\times m_{\text{knn}}}$ is introduced. This matrix is obtained by mapping the values of $S_{v,v}$ in a column vector employing the entries of E_{knn} as row-column pairs to access $S_{v,v}$. Then, a matrix is constructed by repeating the column vector n times along the rows. Notice that this is a valid operation since $S_{v,v}$ is a symmetric matrix. The complete similarity score matrix is obtained as:

$$S = S_{v,e} \odot \bar{S}_{v,v} \odot B \tag{4}$$

where $B \in \mathbb{R}^{n\times m_{\text{knn}}}$ is the oriented incidence matrix and $\odot$ the Hadamard product. The matrix B is used to inject into the scores the knowledge of the edge existence (entries 0) and direction (entries ± 1). i.e. source vertex to target vertex. This information is very helpful since it allows the discrimination of

the edge set based on the sign of their similarity score, i.e. the entries of S. An illustration of the two possible situations that can occur is provided in Fig 4b. The cosine similarity between the sample vertex 30 and its K_{nn} neighbors can provide both positive values, in case the edge direction vectors and the vertex orientation vector of 30 are both in the green region, or negative values if instead they lay in the red region.

Based on the scores contained in the similarity matrix S, a positive and a negative edge for each vertex of $\mathcal{V}$ is sought, being the characterization of an edge as positive or negative related to the sign of the associated score in S. Notice that it may happen that a positive or negative edge for a given vertex does not exist, e.g. in presence of a vertex describing the terminal region of a DLO. In Secs III-C4 and III-C5 the calculus to extract the positive and negative edges from the similarity matrix S of eq. 4 are provided.

4) Positive Edges: Let's define $B_+ \in \mathbb{R}^{n\times m}$ as the positive incidence matrix where the entries -1 of B are set to zero, i.e. B_+ contains values of the set $\{0, +1\}$. Let's also define a row vector $\mathbf{d} \in \mathbb{R}^{1\times m_{\text{knn}}}$ containing the lengths of the edges. A matrix $D \in \mathbb{R}^{n\times m_{\text{knn}}}$ can be created stacking n times $\mathbf{d}$ along the rows. Thus, the entries of D can be filtered out based on B_+ as $D_+ = D \odot B_+$. Then, a generic entry (i, j) of S is weighted based on the associated edge length as $w_+^{ij} = 1 - \frac{d_+^{ij} - \min(\mathbf{d}_+^{\mathbf{i}})}{\max(\mathbf{d}_+^{\mathbf{i}})}$. The vector $\mathbf{d}_+^{\mathbf{i}}$ denotes the i-th row of D_+. The matrix containing all the computed weights is denoted as $W_+ \in \mathbb{R}^{n\times m_{\text{knn}}}$. The presence of B_+ makes W_+ sparse since only the entries associated to an entry $+1$ in B_+ will have a weight different from zero. It follows that $S_+ = S \odot W_+$, where S_+ is the similarity matrix associated to the positive incidence matrix. Finally, an edge, if it exists, is selected for each row of S_+ as the edge associated to the maximum entry of S_+ along the considered row. Thus, considering the generic vertex i, i.e. row i of S_+, its positive edge e_+^i is obtained as $e_+^i = \{E_{\text{knn}}\}_{j^*}$, with $j^* = \text{argmax}(\mathbf{s}_+^{\mathbf{i}})$, $s_+^{ij^*} > \mu$, where with $\mathbf{s}_+^{\mathbf{i}}$ we denote the i-th row of S_+, with $\{E_{\text{knn}}\}_{j^*} \in \mathbb{R}^{1\times 2}$ the column vector at index j^* containing the indices of the source and target vertices and with μ a small threshold to avoid selecting edges with a very low similarity score.

5) Negative Edges: Following a similar discussion to the one of Sec. III-C4, let's define $B_- \in \mathbb{R}^{n\times m}$ as the negative incidence matrix where the entries $+1$ are set to zero, i.e. B_- contains values $\{-1, 0\}$. The entries of D can be filtered out based on B_- as $D_- = D \odot B_-$. The weight matrix associated to D_- can be defined as $W_- \in \mathbb{R}^{n\times m_{\text{knn}}}$ where only the entries associated to -1 in B_- are different from zero. A generic entry w_-^{ij} of W_- is obtained as $w_-^{ij} = 1 - \frac{d_-^{ij} - \min(\mathbf{d}_-^{\mathbf{i}})}{\max(\mathbf{d}_-^{\mathbf{i}})}$. It follows that $S_- = S \odot W_-$, obtaining S_- as the similarity matrix associated to the negative incidence matrix. Finally, an edge, if it exists, is selected for each row of S_- as the edge associated to the minimum entry of S_- along the considered row. The generic edge e_-^i is obtained as $e_-^i = \{E_{\text{knn}}\}_{j^*}$, with $j^* = \text{argmin}(\mathbf{s}_-^{\mathbf{i}})$, $s_-^{ij^*} < -\mu$.

6) Edge Set: The edges obtained from Secs. III-C4 and III-C5 are combined into a single edge set denoted as $\mathcal{E}$ with which the graph $\mathcal{G}$ is generated (Fig. 4c).

Algorithm 1: Intersections Processing

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: $\mathcal{G}'$

1 $\mathcal{V}_{\text{int}} \leftarrow \{v \in \mathcal{V} : \deg(v) > 2\}$
2 $\bar{\mathcal{N}} \leftarrow \emptyset$
3 **foreach** $v \in \mathcal{V}_{int}$ **do**
4 $\quad \mathcal{N} \leftarrow \text{neighbors}(v)$
5 $\quad \bar{\mathcal{N}} \leftarrow \bar{\mathcal{N}} \cup \mathcal{N}$
6 $\bar{\mathcal{N}} \leftarrow \text{merge\_overlapping}(\bar{\mathcal{N}})$
7 $\mathcal{E}_{\text{new}} \leftarrow \emptyset$
8 **foreach** $\mathcal{N} \in \bar{\mathcal{N}}$ **do**
9 $\quad k_{\text{conn}} = |\mathcal{N}| \ \mathbf{div}\ 2$
10 $\quad \mathcal{C} \leftarrow \text{combinations}(\mathcal{N}, 2)$
11 $\quad \mathcal{Z} \leftarrow \text{edge\_solver}(C)$
12 $\quad \mathcal{Z} \leftarrow \text{sorted}(\mathcal{Z}, \text{"descending"})$
13 $\quad \mathcal{V}_{\text{done}} \leftarrow \emptyset,\ c \leftarrow 0$
14 $\quad$ **while** $c \leq k_{conn}$ **do**
15 $\quad\quad$ **foreach** $(v_i, v_j, s_{ij}) \in \mathcal{Z}$ **do**
16 $\quad\quad\quad$ **if** $v_i \notin \mathcal{V}_{done}$ *and* $v_j \notin \mathcal{V}_{done}$ **then**
17 $\quad\quad\quad\quad \mathcal{E}_{\text{new}} \leftarrow \mathcal{E}_{\text{new}} \cup (v_i, v_j)$
18 $\quad\quad\quad\quad \mathcal{V}_{\text{done}} \leftarrow \mathcal{V}_{\text{done}} \cup \{v_i, v_j\}$
19 $\quad\quad\quad\quad c \leftarrow c + 1$

D. Intersections Processing

Although the graph $\mathcal{G}$ should contain vertices having a degree, i.e. number of neighbors, of only 1 or 2, depending on if the considered vertex is an endpoint, vertices having a higher degree, i.e. 3 or more, are still possible. This happens if the considered vertex is placed at the intersection area between multiple DLOs resulting in several ambiguous edge connections, e.g. Fig. 4d. To address this problem, Alg. 1 is employed: it detects the problematic vertices, extracts subgraphs around each of them, and by employing the cosine similarity approach it finds the correct edges.

With more details, Alg. 1 takes as input the graph $\mathcal{G}$ just created and provides as output the updated graph $\mathcal{G}'$ where the ambiguous vertices are removed and their edges redistributed correctly in their local subgraphs. First, the ambiguous vertices are detected as those vertices with a degree larger than 2 and collected in $\mathcal{V}_{\text{int}}$, line 1. Then, for each v in $\mathcal{V}_{\text{int}}$, the neighbor vertices are collected (lines 2 to 5). In case one or more vertices of one set of neighbors overlaps with another one, those sets are merged (line 6) grouping all vertices and treating the problematic area as the composition of the original ones. Each set $\mathcal{N}$ of $\bar{\mathcal{N}}$ defines a subgraph around the problematic area. For each subgraph defined by the vertices in $\mathcal{N}$, the number of connections (edges) to establish is determined by k_{conn} as the integer division between the cardinality of $\mathcal{N}$ and 2 (line 9). The combinations of 2 elements of the vertices contained in $\mathcal{N}$ are collected in the set $\mathcal{C}$ (line 10). These tuples of elements can be considered as edge candidates for the subgraph. For instance, in Fig. 4e, the candidate edges of the subgraph under analysis are depicted in red (wrong) and green (valid). Thus, an edge solver (line 11) is employed

to assign a score to each of those. In particular, given two sample vertices, i.e. v_1 and v_2, which connection should be scored, the direction of the edge connecting them is computed as $\mathbf{d_e^{1,2}} = v_1 - v_2$. Then, the connection cosine similarity score, similarly to eq. (1), is evaluated as

$$s_{\text{int}}(\mathbf{d_v^1}, \mathbf{d_v^2}, \mathbf{d_e^{1,2}}) = \left| s(\mathbf{d_v^1}, \mathbf{d_e^{1,2}})\, s(\mathbf{d_v^2}, \mathbf{d_e^{1,2}}) \right|$$

where with $\mathbf{d_v^1}$ and $\mathbf{d_v^2}$ the vertices orientations are denoted. Notice that the absolute value of the similarity is employed since we are not interested in its sign, but only in its magnitude. Each (v_i, v_j) of $\mathcal{C}$ is therefore augmented by the computed score s_{ij} as (v_i, v_j, s_{ij}) and collected by the set $\mathcal{Z}$ which is then sorted based on the score values in descending order (line 12). Finally, an interactive procedure takes place to loop through the elements of $\mathcal{Z}$ and collect the k_{conn} new edges into $\mathcal{E}_{\text{new}}$ as those defined by vertices not being already assigned to other edges (lines 13 to 19). The sample subgraph analyzed through this paper is solved obtaining the final graph depicted in Fig. 6a.

E. DLOs Instances Extraction

The single instances of the DLOs present in the scene are retrieved considering the connectivity of the graph, i.e. each DLO is represented as an isolated sub-graph from the initial global graph. For each subgraph, the path from one endpoint (vertex with degree 1) to the other is extracted. A path $\mathcal{P}_t$ can be denoted as an ordered sequence of vertices as $\mathcal{P}_t = \{v_1^t, v_2^t, \dots v_{t_n}^t\}$. The extracted path denotes the sequence of key-points describing the DLO instance. From these key-points, a spline curve can be fitted to better approximate the DLO shape and then an estimate of the DLO thickness can be obtained from the distance transform mask M_{dist}. Thus, a colored mask M_c can be drawn as shown in Fig. 6.

In some cases, it can happen that two or more DLO instances are effectively denoted by a single path. This situation can occur in case, for instance, the intersection between two DLOs happens along the border of the image. *RT-DLO*, employing only the mask image, tries to solve this scene by connecting jointly the two distinct DLOs, see as an example Fig. 5 showing the obtained DLOs instances given the source image and mask. To handle this condition, as a final consistency check along the obtained path, the cosine similarity is computed between each vertex of the path and its two neighbors. In particular, given a sample vertex v_i^t, $i \in [2, t_n - 1]$ belonging to path $\mathcal{P}_t$. Its two neighboring vertices are v_{i-1}^t and v_{i+1}^t while the two edges directions are $\mathbf{d_e^{i,i-1}}$ and $\mathbf{d_e^{i,i+1}}$. According to eq. (1), the cosine similarity between $\mathbf{d_v^i}$ and $\mathbf{d_e^{i,i-1}}$ can be denoted as $s_{i,i-1} = s(\mathbf{d_v^i}, \mathbf{d_e^{i,i-1}})$, where $\mathbf{d_v^i}$ describes the orientation of vertex v_i^t. Similarly, $s_{i,i+1} = s(\mathbf{d_v^i}, \mathbf{d_e^{i,i+1}})$. If the product $s_{i,i+1}\, s_{i,i-1}$ is negative, it means that the path is not smooth at vertex v_i^t. Thus, the path $\mathcal{P}_t$ is detached at vertex v_i^t into two different paths, see Fig. 5.

F. Intersections Layout

To correctly assign the DLOs IDs in the intersection areas among two or more DLOs, additional color information is required. Indeed, only from the binary mask M_b and the corresponding constructed graph, this information is not achievable. In this work, we deploy the approach first described in [9]: the standard deviation of the RGB color along the edge connecting two vertices in the area of the intersection is used. For a given intersection, all the involved edges are collected and the standard deviation of the RGB values along the edges compared. The edge corresponding to the smallest value is selected as the one being at the top of the pile. Therefore, the mask M_c is drawn taking into account this information.

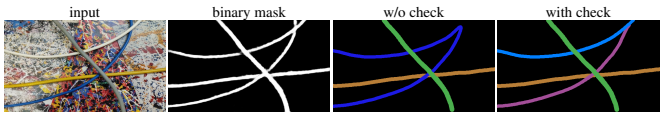


Fig. 5: DLOs instances extraction with and without consistency check in case of a problematic mask.

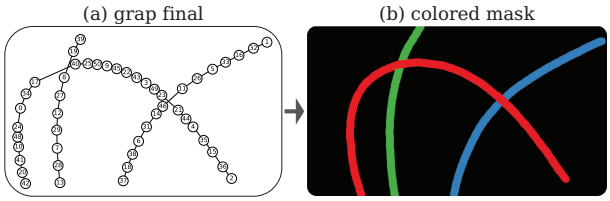


Fig. 6: The connectivity graph (a) is processed to extract the DLOs instances and obtaining the colored mask M_c (b).

IV. Experimental Validation

The experiments were performed employing a workstation with an Intel Core i9-9900K CPU clocked at 3.60GHz and an NVIDIA GeForce GTX 2080 Ti. PyTorch 1.4 is used for software implementation.

A. Test Dataset and Metrics

To evaluate the *RT-DLO* performances on real data, a *test set* originally deployed in [8] and extended in [9] is used. It consists of 135 manually labeled real images of electrical wires with varying diameters and grouped into 3 categories, each consisting of 45 images defining a specific scenario, labelled as *C1*, *C2* and *C3*. Each category is further divided into subclasses based on the number of intersections present in the images, i.e. 1, 2, and 3, with 15 samples each.

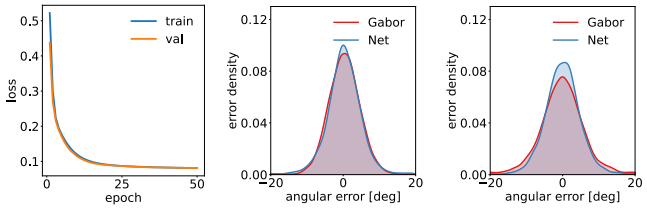
As evaluation metric, the Intersection over Union (IoU $= \frac{|M \cap M_{gt}|}{|M| + |M_{gt}|}$, where M is the mask under evaluation and M_{gt} is the ground truth) is employed. The mask M corresponds to the colored mask M_c where each DLO instance is denoted by a unique color and the IoU score is just the average score across the instances of the image.

B. Training

The training dataset and the training details for the semantic segmentation network employed in Sec. III-A are those of [9]. As the threshold for the segmentation mask M_b, the value of 0.3 is used for its binarization based on [9].

Concerning the CNN network of Sec. III-B2, the dataset was obtained from the synthetic dataset of [9] by randomly

This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2023.3245641



(a) losses (b) eval synthetic (c) eval real

Fig. 7: Evaluation of the CNN angular prediction network and comparison against baseline approach based on Gabor filters. (a) training and validation losses, (b) evaluation on synthetic dataset, (c) evaluation on real dataset. The error density is recovered via a kernel density estimation.

cropping a square patch along the vertices, obtained applying Sec. III-B, and by using the knowledge of the 2D ground truth curve to label the orientations. A patch size $\delta = 15$ was used. The convolutional layers have 32 filter channels as opposed to the 180 neurons for the last linear layer. The network was trained for 50 epochs, employing a batch size of 32 and a learning rate equal to 5×10^{-4}. Adam was selected as optimizer with the final network weights selected based on the validation loss. In Fig 7a the training and validation loss curves smooth decay can be observed, validating the choice of the smooth angle labeling approach.

C. Angle Prediction Evaluation

The network employed to sample the vertices orientations (Sec. III-B2) is compared to a baseline method and tested both on a synthetic test set (100 samples like those of Sec. III-A) and on the real *test set*. As baseline method, an approach based on Gabor filters [22] is used. A Gabor filter is a linear filter usually employed for texture analysis. By properly defining its main parameters, it is possible to obtain a patch similar to the one processed by the network. Thus, the baseline approach consists in: generating 180 Gabor filters spanning $[0, 180[$ degrees; finding the filter with the smallest cumulative difference with respect to the input local patch; assigning as angle prediction the angular value used to generate the filter.

The ground truth angular value for each vertex is directly available in the synthetic data. In the real *test set*, instead, it is recovered from the ground truth instances mask: spline curves are fitted for each instance and the vertices' reference orientation extrapolated as tangent of the curve at a vertex position.

Overall, the proposed network approach shows better performances, especially in the real scenario (see Figs. 7b and 7c). Indeed, we discovered that the Gabor filter approach is more sensitive to the mask's noisy edges and to its characterizing parameters. Considering the real scenario, the error distributions are characterized with the following mean and standard deviation statistics: $-0.03° \pm 6.00°$ for network; $-0.17° \pm 7.07°$ for Gabor.

D. Parameters Choice and Influence

RT-DLO employs two user-defined parameters that can affect the method performances, the vertex sampling ratio α and the number of K_{nn} nearest neighbors. In Tab. I, the performances of *RT-DLO* on the *test set* are compared by varying α and K_{nn}. *RT-DLO* maintains remarkably strong performances across a wide range of values for α, i.e. between 0.1 and 0.3. On the contrary, selecting α as 0.05 results in a quite reduced number of vertices, hurting the description power of the graph. The selection of K_{nn} is also not critical with a value of 8 already sufficient to reach top performances.

TABLE I: Performances of *RT-DLO* when varying the vertices sampling ratio α and the number of K_{nn} nearest neighbors. In bold the values within 1% distance from the maximum one.

K_{nn}	vertices sampling ratio α					
	0.05	0.1	0.15	0.2	0.25	0.3
4	51.20	75.06	77.10	76.70	75.87	75.32
8	57.98	78.19	**79.80**	**79.20**	77.50	77.38
16	56.68	78.72	**79.91**	**79.86**	78.86	78.83
32	56.31	77.78	**79.42**	**79.13**	78.79	78.25

TABLE II: *RT-DLO* versus baseline methods.

Method	Backbone	Key-points	FPS ↑	Time [ms] ↓	IoU [%] ↑
YOLACT	ResNet-50	✗	44	23	32.15
YOLACT	ResNet-101	✗	32	31	35.25
YOLACT++	ResNet-50	✗	42	24	29.96
YOLACT++	ResNet-101	✗	31	32	29.64
BlendMask	ResNet-50	✗	15	66	15.92
BlendMask	ResNet-101	✗	12	81	21.24
CondInst	ResNet-50	✗	16	62	23.29
CondInst	ResNet-101	✗	13	78	29.24
Ariadne+	ResNet-50	✓	3	354	73.96
Ariadne+	ResNet-101	✓	3	360	76.87
FASTDLO	ResNet-50	✓	23	44	73.89
FASTDLO	ResNet-101	✓	22	46	77.77
RT-DLO	**ResNet-50**	✓	**36**	**27**	**77.65**
RT-DLO	**ResNet-101**	✓	**32**	**31**	**79.91**

E. Baseline Methods

RT-DLO is compared against both DLO-specific and general-purpose instance segmentation methods. To the first group belong the algorithms named *Ariadne+* [8] and *FASTDLO* [9]. Both approaches employ the same segmentation network architecture of the one deployed in Sec. III-A. In particular, the network weights are those of [9], thus allowing a straightforward comparison with [9] and [8].

The general purpose DCNN baselines are: *YOLACT* [10], *YOLACT++* [11], *BlendMask* [13] and *CondInst* [14]. Overall, the same dataset configuration and training details of [9] are used to train the aforementioned nets.

F. Evaluation

The comparison of *RT-DLO* against the baseline methods of Sec. IV-E is presented in Tab. II by means of the IoU score computed starting from the color masks provided as output by each method. The table also provides details about the average inference time, FPS, and key-points availability as output. Overall, *RT-DLO* shows strong performances both in terms of IoU score, i.e. $+2.14\%$ and $+3.76\%$ improvements against *FASTDLO*, i.e. top-performing algorithm, when deploying the same segmentation mask M_b. In particular, *RT-DLO* can provide the same level of performance of *FASTDLO*

This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2023.3245641

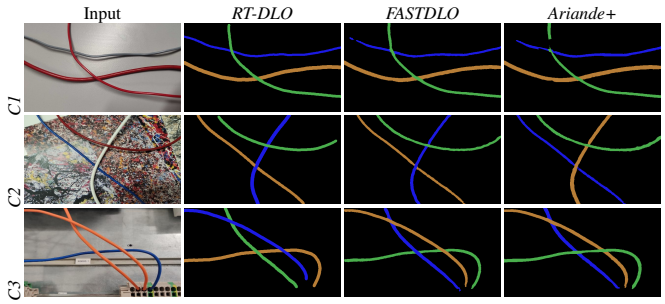


Fig. 8: Qualitative evaluation of *RT-DLO* versus *FASTDLO* and *Ariadne+* on the *test set* classes.

employing a lighter backbone, thus making it possible to reach a frame-rate of 36 FPS, +13 FPS over *FASTDLO*.

A qualitative comparison on a few samples of the *test set* among *RT-DLO* and the DLO-specific methods is provided in Fig. 8, where the superiority of *RT-DLO* is especially visible at the intersections. Indeed, the major advantage of *RT-DLO* against the competing approaches resides in its graph representation which is based on M_b but is less susceptible to degraded area as opposed to the skeleton approach of *FASTDLO* and mask-guided superpixels method of *Ariadne+*. In this regard, a deeper analysis on *RT-DLO* robustness is reported in Sec IV-H, where the requirement of an accurate segmentation mask M_b is experimentally relaxed in two different scenarios: 1) the mask is artificially corrupted with an erosion process; 2) the segmentation back-end specifically trained on electric wires is replaced with others back-ends trained on general purposed datasets.

G. Evaluation of Inference Time

In Tab. III a characterization of the average timing on the *test set* for each stage of the proposed method is provided. Faster processing times can be achieved by deploying a lighter backbone, such as ResNet-50, saving several milliseconds in the binary segmentation phase and obtaining a total processing time of about 27 ms as opposed to 31 ms. The graph generation time is below 10 ms, highlighting the efficiency of *RT-DLO*. If the colored mask is not required, the last two stages can be skipped shortening the computation time by 1 to 2 milliseconds depending on the number of intersections, as shown in Tab. III. The timings in the table are obtained employing the hardware setup highlighted at the beginning of Sec. IV. A similar timing of about 13.5 ms is obtained for the *Total w/o Segmentation* case with a consumer laptop (Intel Core i7-12700H CPU). Indeed, high computation power is mostly required for the deep segmentation network. Thus, if the application does not require a complex deep model for scene semantic segmentation, the hardware specifications can be relaxed or, alternatively, higher overall FPS can be achieved.

H. Mask Degradation and Different Segmentation Back-Ends

The improvements of *RT-DLO* against the main competing methods, i.e. the DLO-specific algorithms *Ariadne+* and *FASTDLO*, are not only in the form of faster processing time and better accuracy. Indeed, an important benefit of the graph representation approach of *RT-DLO* is its ability to better handle degraded semantic segmentation masks M_b. To illustrate the graph-based advantage of *RT-DLO*, a two-fold study is conducted. On one hand, the performance drop of *RT-DLO* and the competition is evaluated after an erosion process is applied on M_b. On the other hand, different segmentation networks trained on public datasets, i.e. not DLO-specific ones, are employed.

Procedure	Number Intersections 1	2	3	avg
Binary Segmentation	19.72	19.49	19.49	19.50
Graph Generation	6.69	8.23	9.17	8.03
Intersections Processing	0.57	0.80	0.95	0.77
DLOs Instances Extraction	1.25	1.55	1.79	1.53
Intersections Layout	0.61	1.13	1.65	1.13
Output Colored Mask	0.37	0.45	0.51	0.44
Total w/o Segmentation	9.50	12.15	14.07	11.91
Total	29.22	31.64	33.56	31.41

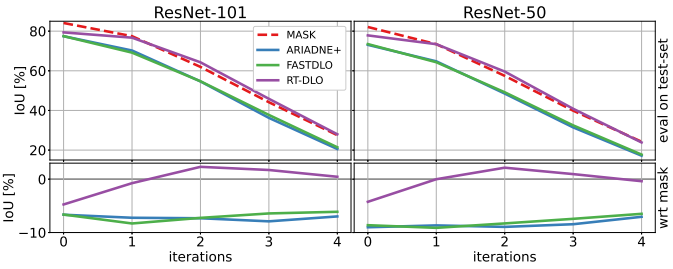
TABLE III: Average execution times [ms] of the main *RT-DLO* stages wrt the number of intersections in the image.

Concerning the first study, the masks M_b of the *test set* are iteratively eroded, that is the process consisting in thinning the foreground area of a binary mask, with a kernel of 3×3 pixels to simulate the effects of less precise masks coming from Sec. III-A. The evaluation is performed by comparing *RT-DLO* to the DLO-specific methods on the masks obtained from the two different backbones, i.e. ResNet-50 and ResNet-101, see Fig. 9. From the plots of Fig. 9a, *RT-DLO* shows the capability of maintaining an almost steady performance after the first round of erosion process, followed by a drop in the scores in the subsequent iterations. On the contrary, the drop of scores associated to *FASTDLO* and *Ariadne+* is significant from the very first iteration. Considering the mask IoU score as an upper bound, *RT-DLO* is capable of maximizing its score as opposed to the compared approaches. The images of Fig. 9b allow to catch better the effects of the erosion process and the *RT-DLO* advantages on the test sample *C1* of Fig. 8.

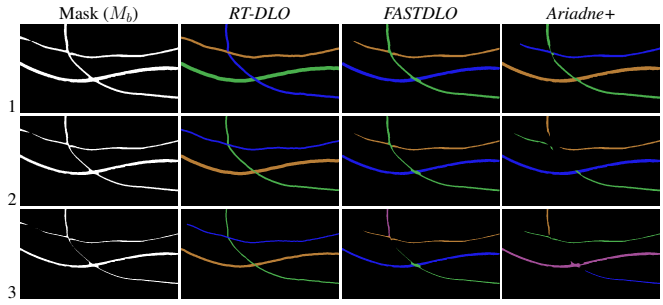
A study about replacing the segmentation back-end of Sec. III-A with ImageNet pre-trained salient object segmentation (SOS) approaches is conducted in Fig. 10, avoiding the need of training a segmentation network on a DLO-specific dataset. The SOS architectures tested are: *EGNet* [23], *F3Net* [24], *CPD* [25] and *PoolNet* [26]. When evaluated on the *test set*, *RT-DLO* continues to achieve strong performances compared to the competing approaches, see Fig. 10a. The advantages of *RT-DLO* in the case of degraded masks are even more apparent for the sample images of Fig. 10b which show how *RT-DLO* is able to minimize the number of extracted instances.

V. Conclusions

In this paper, a novel method for real-time instance segmentation of DLOs is presented. The representation of the DLOs as a graph offers an efficient, simple and intuitive way to obtain the DLOs instances. The segmentation performance improvements compared to current state-of-the-art approaches for DLOs detection are noticeable. More importantly, the inference time capabilities of *RT-DLO* make it stand out

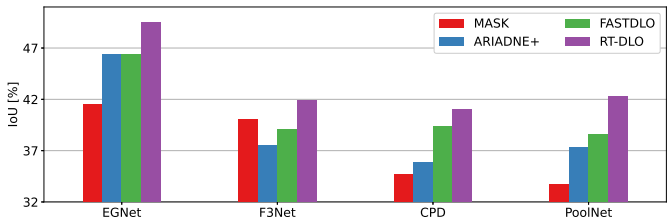


(a) evaluation on the *test set* eroding M_b.

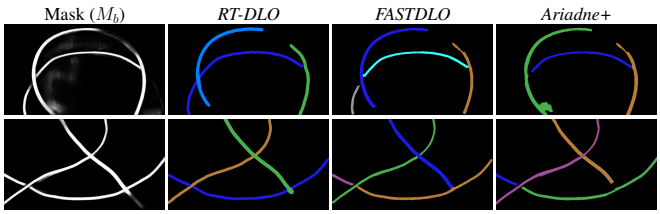


(b) qualitative comparison given M_b eroded for 1, 2 and 3 iterations.

Fig. 9: *RT-DLO*, *FASTDLO* and *Ariadne+* performance comparison after iteratively degrading the binary mask M_b.



(a) evaluation on the *test set* employing M_b obtained by SOS networks.



(b) qualitative comparison of the instances masks given M_b from *EGNet*.

Fig. 10: Comparison of *RT-DLO*, *FASTDLO* and *Ariadne+* when employing popular salient object segmentation networks.

even more compared to existing approaches. In future works, *RT-DLO* can be improved and expanded in several ways. For example, *RT-DLO* currently processes each image individually. However, the segmentation stage can be substituted with a different approach exploiting the previous frames of a video sequence for a better and possibly faster segmentation mask. In this context, a tracking system can be also investigated to match the DLO instances across the video sequence. Finally, the graph-based representation of DLOs can be easily extended to other structures, like wiring harnesses, and other sensors, like 3D cameras resulting in a 3D graph.

REFERENCES

[1] 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," *The International Journal of Robotics Research*, 2018.

[2] K. P. Cop, A. Peters, B. L. Žagar, D. Hettegger, and A. C. Knoll, "New metrics for industrial depth sensors evaluation for precise robotic applications," in *Proc. IEEE/RSJ Int. Conf. IROS*, 2021.

[3] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE robotics and automation letters*, 2020.

[4] N. Lv, J. Liu, and Y. Jia, "Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations," *IEEE Transactions on Robotics*, 2022.

[5] A. Keipour, M. Bandari, and S. Schaal, "Deformable one-dimensional object detection for routing and manipulation," *IEEE Robotics and Automation Letters*, 2022.

[6] X. Jiang, K.-m. Koo, K. Kikuchi, A. Konno, and M. Uchiyama, "Robotized assembly of a wire harness in a car production line," *Advanced Robotics*, 2011.

[7] D. D. Gregorio, G. Palli, and L. D. Stefano, "Let's take a walk on superpixels graphs: Deformable linear objects segmentation and model estimation," in *Asian Conference on Computer Vision*. Springer, 2018.

[8] A. Caporali, R. Zanella, D. De Gregorio, and G. Palli, "Ariadne+: Deep learning-based augmented framework for the instance segmentation of wires," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022.

[9] A. Caporali, K. Galassi, R. Zanella, and G. Palli, "Fastdlo: Fast deformable linear objects instance segmentation," *Robotics and Automation Letters*, 2022.

[10] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE/CVF Conf. ICCV*, 2019.

[11] ——, "Yolact++: Better real-time instance segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE Int. Conf. ICCV*, 2017.

[13] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, "Blendmask: Top-down meets bottom-up for instance segmentation," in *Proceedings of the IEEE/CVF Conf. CVPR*, 2020.

[14] Z. Tian, C. Shen, and H. Chen, "Conditional convolutions for instance segmentation," in *Proceedings of the Conf. ECCV*. Springer, 2020.

[15] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "Solov2: Dynamic and fast instance segmentation," *Advances in Neural information processing systems*, 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. of the ICCCR*, 2021.

[17] Y. Wang, D. McConachie, and D. Berenson, "Tracking partially-occluded deformable objects while enforcing geometric constraints," in *2021 IEEE Int. Conf. ICRA*. IEEE, 2021.

[18] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. European Conf. on computer vision*, 2018, pp. 801–818.

[19] G. Borgefors, "Distance transformations in digital images," *Computer vision, graphics, and image processing*, 1986.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[21] X. Yang and J. Yan, "Arbitrary-oriented object detection with circular smooth label," in *ECCV*. Springer, 2020.

[22] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[23] J.-X. Zhao, J.-J. Liu, D.-P. Fan, Y. Cao, J. Yang, and M.-M. Cheng, "Egnet: Edge guidance network for salient object detection," in *Proceedings of the IEEE/CVF Conf. ICCV*, 2019.

[24] J. Wei, S. Wang, and Q. Huang, "F^3net: fusion, feedback and focus for salient object detection," in *Proceedings of AAAI-20*, 2020.

[25] Z. Wu, L. Su, and Q. Huang, "Cascaded partial decoder for fast and accurate salient object detection," in *IEEE/CVF Conf. CVPR*, 2019.

[26] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang, "A simple pooling-based design for real-time salient object detection," in *IEEE/CVF Conf. CVPR*, 2019.

This article has been accepted for publication in IEEE Transactions on Industrial Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TII.2023.3245641

Alessio Caporali Alessio Caporali received the M.Sc. degree in automation engineering from the University of Bologna, Italy in 2019. He is currently pursuing the Ph.D. degree in biomedical, electrical and system engineering with the University of Bologna. His research interests include computer vision for robotic manipulation of deformable objects.

Kevin Galassi Kevin Galassi received the B.Sc. and M.Sc. degree in automation engineering from the University of Bologna, Italy, respectively in 2018 and 2020. He is currently pursuing the National Ph.D. program in Artificial Intelligence at the laboratory of Robotics (LAR) at the University of Bologna. His research interests include industrial robotic manipulation of deformable objects, collaborative robotics and robotic learning.

Bare Luka Žagar Bare Luka Žagar is a Ph.D. candidate currently working in the Chair of Robotics, Artificial Intelligence and Real-time Systems at the Technical University of Munich (TUM). He completed his B.Sc. and M.Sc. in Control Engineering and Automation at the University of Zagreb (Faculty of Electrical Engineering and Computing). His research interests include computer vision for robotics, point cloud representation learning and 3D object detection.

Riccardo Zanella Riccardo Zanella received the M.Sc. degree in automation engineering from the University of Padova, Italy in 2016, and the Ph.D. degree in biomedical, electrical and system engineering from the University of Bologna, Italy in 2021. He is currently working as Postdoctoral Researcher at the University of Bologna. His research interests include computer vision and robot learning for autonomous manipulation of deformable objects.

Gianluca Palli 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 150 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.

Alois C Knoll Alois C Knoll (Senior Member, IEEE) received the M.Sc. degree in electrical /communications engineering from the University of Stuttgart, Stuttgart, Germany, in 1985, and the Ph.D. (summa cum laude) degree in computer science from the Technical University of Berlin (TU), Berlin, Germany, in 1988. He served on the faculty of the Computer Science Department of TU Berlin until 1993, when he qualified for teaching computer science at a university (habilitation). Following this, he was a Full Professor and the Director of the Computer Engineering research group at the University of Bielefeld until 1991, when he assumed the Professorship of Real-Time Systems and Robotics at Technical University of Munich, Munich, Germany. His research interests include cognitive, medical and sensor-based robotics, multiagent systems, data fusion, adaptive systems, multimedia information retrieval, model-driven development of embedded systems with applications to automotive software and electric transportation, and simulation systems for robotics and traffic.