FASTDLO: Fast Deformable Linear Objects Instance Segmentation

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

***Abstract*—In this paper, an approach for fast and accurate segmentation of Deformable Linear Objects (DLOs) named *FASTDLO* is presented. A deep convolutional neural network is employed for background segmentation, generating a binary mask that isolates DLOs in the image. Thereafter, the obtained mask is processed with a skeletonization algorithm and the intersections between different DLOs are solved with a similarity-based network. Apart from the usual pixel-wise color-mapped image, *FASTDLO* also describes each DLO instance with a sequence of 2D coordinates, enabling the possibility of modeling the DLO instances with splines curves, for example. Synthetically generated data are exploited for the training of the data-driven methods, avoiding expensive collection and annotations of real data. *FASTDLO* is experimentally compared against both a DLO-specific approach and general-purpose deep learning instance segmentation models, achieving better overall performances and a processing rate higher than 20 FPS.**

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

I. Introduction

DEFORMABLE Linear Objects (DLOs) are a special subgroup of deformable objects consisting, among the main constituents, of cables, wires, ropes, suture threads and elastic tubes [1]. Despite DLOs being vastly present both in domestic and in industrial environments, very few robotics systems are currently deployed in scenarios in which a proper perception of DLOs is required. Indeed, the manufacturing and assembly industries working with wires and wiring harness still largely rely on human labor whereas the introduction of robotic solutions is only discussed and studied at a research level, e.g. in automotive [2] and aerospace industries [3]. This deployment gap is due to the lack of a stable, efficient and accurate approach for the perception of this class of objects.

In this paper, an algorithm named *FASTDLO* (FAst Segmentation of Deformable Linear Objects) for a reliable, accurate and fast instance segmentation of DLOs assuming no knowledge about the background and the number of objects in the scene is presented. *FASTDLO* takes as input an image and provides as output a colored mask where each DLO instance is denoted with a unique color. In addition, *FASTDLO* outputs a sequence of key-points for each DLO instance. Thus, it is possible to model the DLO instances by means, for example, of spline curves, providing in this way an important and useful description for robotic manipulation tasks. From the input image, the background, i.e. pixels not corresponding to a DLO-like object, is removed by means of a Deep Convolutional Neural Network (DCNN), generating as output a binary mask. Thereafter, the binary mask is processed with a skeletonization algorithm and the ambiguous intersections between the DLOs are solved with a second data-driven approach based on a shallow similarity-based neural network. Synthetically generated data are deployed in the learning-based methods allowing a fast adaptability to every possible custom scenario. *FASTDLO* achieves a processing rate higher than 20 Frames-Per-Second (FPS) with an image size of 640 $\times$ 360 pixels, employing, as hardware, a workstation with an Intel Core i9-9900 K CPU clocked at 3.60 GHz and an NVIDIA GeForce GTX 2080 Ti. PyTorch 1.4 is used for the software implementation. To summarize, the main contributions of this paper are:

- A reliable and efficient method for the instance segmentation of DLOs in images without assumptions about the type of background and the number of objects in the scene;
- Deployment of synthetic data for all the data-driven approaches involved in the proposed method, enabling a faster adaptability to specific use-cases;
- Exploitation of similarity learning to discern the DLO instances at intersection-level employing both appearance-based and topological features;
- Better overall performances in terms of speed and accuracy compared to existing methods available in the literature, both DLO-specific and general-purpose ones.

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

II. Related Works

The importance of DLOs in a large variety of applications results in a high interest in solutions that allows their correct and precise identification for different tasks such as cable manipulation [4] for switchgears and harnesses manufacturing or DLOs shape estimation [5] by means of multiple 2D images.

Manuscript received 24 February 2022; accepted 20 June 2022. Date of publication 15 July 2022; date of current version 20 July 2022. This letter was recommended for publication by Associate Editor Z. Min and Editor C. Cadena Lerma upon evaluation of the reviewers' comments. This work was supported by the European Commission's Horizon 2020 Framework Programme through Project REMODEL - Robotic technologies for the manipulation of complex deformable linear objects - under Grant Agreement 870133. *(Corresponding author: Alessio Caporali.)*

The authors are with the DEI - Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: alessio.caporali2@unibo.it; kevin.galassi2@unibo.it; riccardo.zanella2@unibo.it; gianluca.palli@unibo.it).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3189791, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3189791

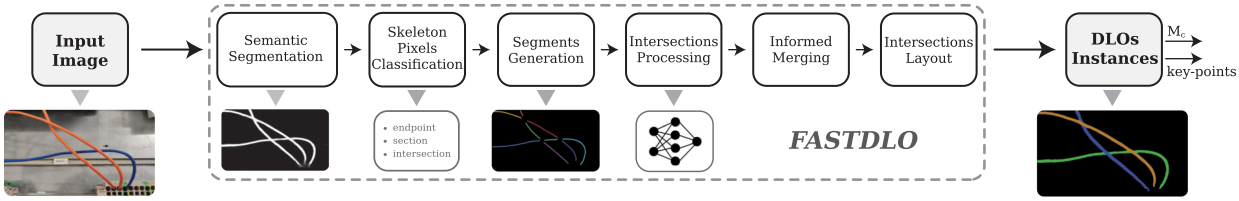


Fig. 1. The *FASTDLO* algorithm.

In the past, the problem of DLO identification has been solved in simple settings: in [6] the authors required the presence of a single DLO in the scene and its segmentation was based on a color threshold with a controlled background; in [7] a good contrast between the background and the DLO is again assumed; in [8] a threshold is again applied in a controlled background to segment the cable. Indeed, the major difficulties in DLO identification rely on its simplicity, which does not offer distinctive features to be used for an unambiguous detection. Moreover, the approach proposed in [6] assumes to deal with just a single DLO in the scene. These assumptions about segmentation capability and number of expected instances limit the applicability of the proposed solutions in real-case scenario where DLOs are commonly involved. On the contrary, *FASTDLO* does not make any assumption about the background and the number of DLOs in the scene.

Concerning advanced DLO-specific approaches, the earliest contribution for DLO detection, segmentation and modeling is represented by *Ariadne* [9], an algorithm based on the over-segmentation of the source image employing superpixels, developed to perform DLO segmentation in case of complex backgrounds. *Ariadne+* [10] was recently introduced as an improved version of *Ariadne* concerning several aspects: better accuracy and efficiency, ability to consider even more complex scenarios in which the endpoints of the cable were not present in the image. Currently, *Ariadne+* represents the state of the art in terms of DLOs instance segmentation. However, its throughput is limited to a few FPS providing a strong limiting factor for its applicability on real-world applications. In this regard, instead of a paths discovery method based on the superpixelization that requires significant processing effort and the definition of the number of superpixels in an image, *FASTDLO* focuses directly on solving the intersection areas of the image between multiple DLOs to distinguish the instances, increasing the speed and accuracy of the results. Indeed, these improvements can be beneficial for DLO tracking problems that have been only relatively solved in partially occluded environments [11], in simulation [12] or with markers attached on the DLO [2], [13].

Regarding other data-driven approaches, the advancements of deep learning in the last years resulted in several DCNN tailored for the general problem of instance segmentation task, e.g. [14]–[17]. In addition, the segmentation of wires and cables via learning-based methods has been attempted in [18] where a dataset consisting of electric wires obtained with a chroma-key approach is made publicly available. A relevant problem in the application of data-driven approaches for the instance segmentation of DLOs resides in the lack of good-quality publicly available datasets and, consequently, the difficulty in annotating a large set of images. However, some approaches are emerged focusing on synthetic data generation pipelines [19], [20] that tackle this problem.

For the sake of completeness, other methods exist that rely on different sensing approaches and instead of images use other sensors for the detection of DLOs such as electrical cables, e.g. sensorized tactile fingers in [21]. These methods ca be useful in case of occlusions affecting the camera view due to tight operating spaces.

III. The *FASTDLO* Algorithm

The *FASTDLO* pipeline, schematized in Fig. 1, consists of the following main steps:

A) *Background Segmentation:* A DCNN performs the segmentation of the source image discerning background pixels from DLOs pixels, outputting a binary mask M_b.
B) *Skeleton Pixels Classification:* A skeleton M_s is generated from the mask M_b and its pixels are classified depending of their local neighborhoods.
C) *Segments Generation:* The intersection areas of M_s are filtered out and *segments* are generated;
D) *Intersections Processing:* A shallow neural network is employed to predict connection probabilities among endpoint-pairs;
E) *Informed Merging:* The *segments* are concatenated to recover the full description of each DLO employing the result of intersections processing;
F) *Intersections Layout:* The standard deviations of the DLOs instances RGB colors at intersections-level are used to asses the correct ordering at the intersection areas to create correct instance masks.

The aforementioned steps are deeply analyzed in the following.

A. Background Segmentation

The generic input image I_s is processed by means of a DCNN performing the semantic segmentation, i.e. the task of labeling each pixel of an image with a given class. For the paper purposes, just one class (the DLO) is defined, thus the output of the segmentation is simply a binary mask M_b with the DLOs pixels labeled in white.

Aiming toward the prediction of a reliable binary mask, and due to the usual difficulties in data collection and labeling for deep learning applications, a novel pipeline [19] making use

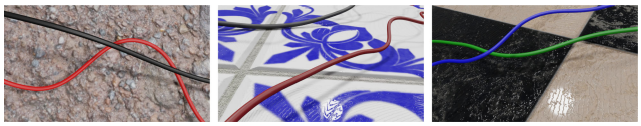


Fig. 2. Synthetically generated images used during training.

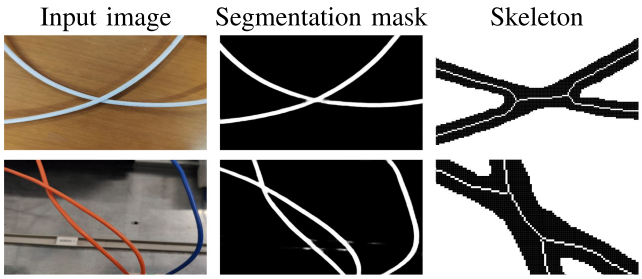


Fig. 3. Input images with background segmentation results and generated skeletons zoomed at the intersections area. To clarify the skeleton visualization, the mask colors in the right column are inverted.

of Blender to render realistic images is exploited. A dataset of synthetically generated cables is built randomizing the shapes, radius, color and stripes of the DLOs. A random texture is chosen as background and the scene lighting conditions are randomized as well creating different combinations of shadows. All these expedients are needed to enhance the generalization capabilities of the network during training. Overall, a total of about 32,000 images were rendered to handle the data-driven learning methods involved throughout this paper. As an example, in Fig. 2 some generated images are shown.

As network architecture for the background segmentation, DeeplabV3+ [22] is selected since it provides reliable performances in the context of DLOs especially along object boundaries, as demonstrated also in [10], [18].

In Fig. 3 an example of the segmentation process on real samples is shown: the shadows present in the input image do not affect the predicted mask and are successfully neglected; instead, the background object in the second row is more difficult to handle since it appears as a thin wire in the image, so few false positives can be found in the associated mask.

B. Skeleton Pixels Classification

The segmentation mask M_b is processed with a skeletonization algorithm consisting of a thinning iterative approach which erodes the input mask. Thus, a new mask M_s is obtained having the following properties: 1) same connectivity as the input mask; 2) 1-pixel width across the mask instead of the original mask thickness; 3) equidistant skeleton to the borders of M_b. In Fig. 3 (last column) the masks M_b and M_s are combined to highlight these properties.

From an image comprising several DLOs and exploiting the linearity property of the latter, for each pixel of the skeleton M_s, defining a small (3×3) kernel, only three types of local neighbors can be experienced. They are depicted in Fig. 4 with the target pixel at the center of the local region highlighted with a red contour. After the skeletonization, each pixel of M_s receives a label depending on its local neighborhood:

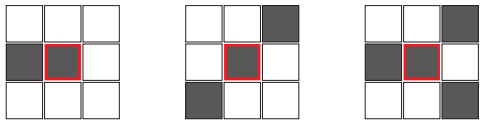


Fig. 4. Local neighbors possibilities of a skeleton pixel given a 3×3 kernel. To clarify the representation, the skeleton is in dark while the background is in white.

- *endpoint:* only one pixel in addition to the central one is contained in the local neighborhood, i.e. Fig. 4(a);
- *section:* two more pixels are present in the neighborhood, i.e. Fig. 4(b). The term *section* refers to the considered pixel being placed along a section of a DLO and not at its end.
- *intersection:* the central pixel is surrounded by three more pixels in the neighborhood forming, in general, a characteristic 'Y' shape [6], i.e. Fig. 4(c). This condition in case of binary masks describing DLOs occurs when two DLOs cross each other.

C. Segments Generation

The *intersection* pixels with their surrounding area are discarded from M_s since they correspond to a topologically misleading region of M_s due to the DLOs crossing. Indeed, these phenomena can be appreciated in Fig. 3 where the generated skeletons nearby the intersection pixels do not describe correctly the DLO topology, i.e. center line, as opposed to the skeleton pixels far away from it. The discard operation is performed based on the distance transform image of the local area considered. The distance transform is an operation that computes the distance, in pixel values, between a given pixel location to the nearest boundary [23], i.e. black pixels of M_b.

Because of the removal of the intersection areas, new *endpoint* pixels emerge in the updated skeleton. Thus, *segments* are generated between two connected *endpoints*. A *segment* is defined as an ordered sequence of pixels where the elements inside the sequence are *sections* whereas the extremities are *endpoints*. The *segment* sequence can be effectively obtained by sliding the skeleton with a 3×3 kernel from one of its *endpoints*, collecting the only pixel not already in the *segment* under construction and updating the kernel anchor to the added pixel location. In addition, for each *segment*, a common thickness is estimated based on a distance transform previously computed. The overall *segment* thickness is obtained by computing the median value of the distances gathered for each *segment's* pixel. The median allows gaining robustness against spurious values due to noisy boundaries in M_b. In Fig. 1 the *segments* generated for the considered image are denoted with unique colors.

D. Intersections Processing

The intersections among the DLOs are solved by comparing the feature vectors of the endpoints of two candidate segments via a shallow neural network, i.e. similarity network, predicting

Algorithm 1: Intersections Processing.

Input: $\mathcal{C}$, $\mathcal{S}$, $\mathcal{E}$
Output: $\mathcal{Z}$

1 $\mathcal{P} \leftarrow \emptyset$ // endpoint-pairs collection
2 **foreach** $c \in \mathcal{C}$ **do**
3 $\quad \mathcal{E}_c \leftarrow \{e \in \mathcal{E} \cap e \in \mathcal{C}\}$
4 $\quad \mathcal{P}_c \leftarrow combination(\mathcal{E}_c, 2)$
5 $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_c$
6 $\mathcal{Z} \leftarrow \emptyset$ // similarity network predictions
7 **foreach** $(e_i, e_j) \in \mathcal{P}$ **do**
8 $\quad x_i \leftarrow getFeatureVector(e_i)$
9 $\quad x_j \leftarrow getFeatureVector(e_j)$
10 $\quad z_i \leftarrow computeEbbendingVector(x_i)$
11 $\quad z_j \leftarrow computeEbbendingVector(x_j)$
12 $\quad p_{ij} \leftarrow e^{-\|z_i, z_j\|^2}$
13 $\quad \mathcal{Z} \leftarrow \mathcal{Z} \cup \{e_i, e_j, p_{ij}\}$
14 return $\mathcal{Z}$

the probability of their connection. The computation of the connection probabilities is schematized in Algorithm 1 showing the two main phases: endpoint-pairs collection; similarity network predictions. The inputs of the algorithm are the set of all the *intersections* in the image, i.e. $\mathcal{C}$, and, given the updated skeleton of Section III-C, the sets of the *endpoints* and of the *segments*, i.e. $\mathcal{E}$ and $\mathcal{S}$ respectively.

The approach starts by collecting all the endpoint-pairs which connection needs to be evaluated by initializing $\mathcal{P}$ empty (line 1). Thus, for each intersection to be solved c, the endpoints of the *segments* associated to c, i.e. originally connected to c before removing the *intersection* pixels from the skeleton (see Section III-C), are extracted from $\mathcal{E}$ and collected into $\mathcal{E}_c$, line 3. Then, the components of $\mathcal{E}_c$ are organized into combinations of 2 elements, i.e. endpoint-pairs, in $\mathcal{P}_c$, and the set of endpoint-pairs $\mathcal{P}$ is updated accordingly, lines 4 and 5.

The collected endpoints-pairs $\mathcal{P}$ are now processed by a similarity network. The goal of this network is to transform an input feature vector into an embedding space where similar input vectors are close together and dissimilar ones are far apart. In the setup adopted in this work, the triplet loss [24] is deployed for the required optimization of the network. The loss is computed between an anchor, a positive and a negative sample. The distance in the embedding space between anchor and positive is minimized, while the one between anchor and negative is maximized. The input feature vectors are obtained from the endpoints of the segments around a given intersection. As feature elements of the input vector $x \in \mathbb{R}^{d_i}$, the following values are used: RGB color of the local endpoint area; thickness of the segment associated to the endpoint; endpoint direction estimate.

In Algorithm 1, the feature vector for the endpoints e_i and e_j are created at lines 8 and 9. Then, a forward pass in the network layers is performed to compute the embedding vectors z_i and z_j, lines 10 and 11.

As mentioned, the prediction is based on the distance of the embedding vectors which can be computed as $d_{ij} = \|z_i, z_j\|^2$, where $\|\cdot\|^2$ denotes the L2-distance. To obtain a probability-like value in the [0,1] range describing the likelihood of the connection, the distance is transformed by means of a Gaussian activation function as $p_{ij} = e^{-d_{ij}}$. This last step in the similarity network is provided at line 12 while an illustration schematizing the computation flow of the similarity network from its inputs till the predicted connection score is available in Fig. 5.

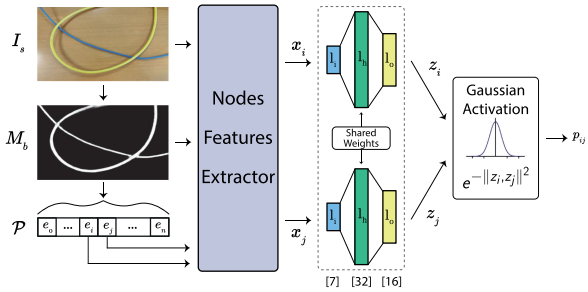
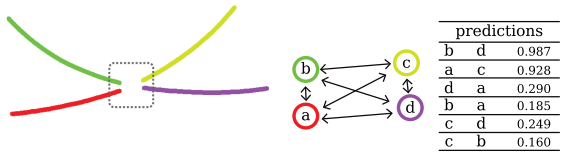


Fig. 5. Endpoint-pair probability computation. For the general endpoint e_i, from the source image I_s and binary mask M_b a feature vector is created as x_i. The embedding vector z_i is obtained after the propagation of x_i in the similarity network layers. A Gaussian activation function on the embeddings L2-distance is employed for calculating the final score p_{ij} of the endpoints e_i and e_j.



predictions		
b	d	0.987
a	c	0.928
d	a	0.290
b	a	0.185
c	d	0.249
c	b	0.160

(a) different *segments* (b) intersection processing with predictions of the endpoints pairs

Fig. 6. (a) *segments* generated; (b) example of intersection processing of the region highlighted area in (a).

To conclude, for each endpoint-pair a probability value p_{ij} is computed and the set $\mathcal{Z}$ updated (line 13) with tuples of three values, i.e. endpoint-pair (e_i, e_j) and connection probability p_{ij}. Although Algorithm 1 describes the process for each individual element of $\mathcal{P}$, the actual implementation is based on batch processing enabling an efficient computation of the scores, as described in Section IV-F. In Fig. 6(b) an example of the processing for four segments (six endpoint-pairs) extracted from Fig. 3 (first row) is shown.

E. Informed Merging

Exploiting the endpoint-pairs connection probabilities computed in Section III-D it is possible to concatenate *segments* obtaining the full description of each DLO in the image. This concatenation process is addressed as informed merging and it is schematized in Algorithm 2, showing how $\mathcal{Z}$ is employed to iteratively update $\mathcal{S}$ till each $s \in \mathcal{S}$ describes a whole DLO.

First, the elements of $\mathcal{Z}$ are sorted based on the connection probability values in descending order (line 1), thus prioritizing during the merging process the most probable connections. The set of nodes already processed, i.e. $\mathcal{E}_z$, is initialized to zero at line 2. An iteration on the elements of $\mathcal{Z}$ is performed and, starting from the highest score and moving toward the lowest one, the merging of the segments, i.e. Fig. 6(a), is executed. Indeed, if both the endpoints retrieved from one of the elements of $\mathcal{Z}$ are not already processed (line 4), and their endpoint-pair

Algorithm 2: Informed Merging.

Input: $\mathcal{Z}$, $\mathcal{S}$
Output: $\mathcal{S}$

```
1  Z ← sorted(Z, "descending")
2  E_z ← ∅
3  foreach (e_i, e_j, p_ij) ∈ Z do
4      if e_i ∉ E_z and e_j ∉ E_z then
5          if p_ij > t_c then
6              s_i ← getSegmentFromEndpoint(S, e_i)
7              s_j ← getSegmentFromEndpoint(S, e_j)
8              s ← s_i ∪ s_j
9              S ← S ∪ s \ {s_i, s_j}
10             E_z ← E_z ∪ {e_i, e_j}
11 return S
```

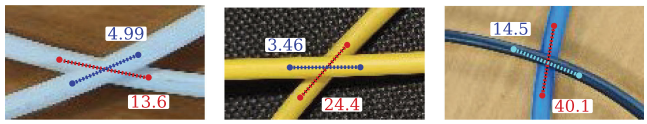
connection probability is larger than a user-defined threshold t_c (line 5), the two corresponding *segments* associated to the endpoints are collected (lines 6 and 7), merged together (line 8) and the *segments* set updated (line 9). With the term *informed* merging we refer to the high-level operation of performing the union between the two *segments* sets taking into consideration their ordering, e.g. head-tail, tail-tail and all the others combinations.

Consequently, the endpoint-pairs having lower scores and with one of the two endpoint elements already associated are not considered and their merging avoided. The describes association continues for all the elements of $\mathcal{Z}$ having a connection probability larger than the threshold t_c, introduced to avoid merging endpoints with incompatible orientations, colors or thicknesses. This threshold is effectively used only in situations where the mask M_b is not reliable and edge-conditions occur. Instead, in normal settings, the merging process would result in first high probability endpoints association thus making the low probability ones already incompatible irrespective of the threshold value.

The presented merging process is performed directly on the existing *segments*, thus the operation is propagated by updating the set of *segments* accordingly. For instance, in case of a *segment* disputed by two different intersections, at the second merging process the operation of joining the two candidates *segments* is performed on the new merged *segment* (obtained after the first merging process) and not on the initial one.

F. Intersections Layout

As additional information aiming at providing a complete and accurate solution of the scene, the order of the DLOs in a given intersection, i.e. which is the one at the top of the pile, is provided by comparing the standard deviation of the RGB colors along the line connecting the endpoint-pair previously solved. For example, given an intersection made of two DLOs, i.e. with four endpoints, and hence two endpoint-pairs predicted, the RGB color values along the two endpoint-pairs positions are collected and their mean standard deviation, i.e. the mean of the standard deviations computed for each channel, compared. The pair with the smallest standard deviation is assumed to be at the top of the intersection pile, while the highest standard deviation pair below. The difference in the value is due to the change in the color along the line for the DLO not at the top or, in case of DLOs with identical colors, mostly due to the shadows projected from the above DLO onto the one below. In the case of a cross composed of three or more DLOs, only the instance above them all can be identified since the continuity in the colors in the intersection region is the main deciding condition. This continuity is only met for the top DLO. The approach is quite simple and yet proves to be effective and inherently fast given the intersection solution provided in Section III-D is reliable, more in Section IV-A.



(a) example 1 (b) example 2 (c) example 3

Fig. 7. Example of the intersection layout estimation with DLOs having identical colors, in (a) and (b), and different colors (c). In all examples the DLO at the top is blue labeled.

DLOs ordering in an intersection is particularly needed in case this approach is integrated into a larger manipulation pipeline with a robotic system for routing or pick and place tasks. The information about the layout of the DLOs in an intersection is missing both in [9] and [6]. Instead, in [10] a solution based on a data-driven classification approach is presented requiring specific training and constrained at precise image crop resolutions. On the contrary, the approach presented in this paper exploits the accurate DLOs center line localization obtained from the skeleton, as opposed to the superpixels centroids used in [10], avoiding then the introduction of additional data-driven approaches. In Fig. 7 some example intersections are displayed with the computed values.

IV. Experimental Validation

A. Training

The training dataset is obtained from 90% of the synthetic dataset described in Section III-A, while the validation from the remaining 10%. The segmentation network of Section III-A, i.e. [22], employed in *FASTDLO* is trained with a ResNet-101 [25] backbone pre-trained on ImageNet for 250 K iterations with the final weights selected as the ones corresponding to the lowest validation loss. As hyper-parameters, we employed a batch size of 10, output stride of 16, separable convolutions, Adam as optimizer and a polynomial learning rate adjustment policy with power 0.95 starting from 10^{-6} to a minimum of 10^{-9}. As augmentation scheme we deploy: channel shuffling; hue, saturation and value randomization; flipping; perspective distortions; random cropping; random brightness and contrast.

Concerning the similarity network of Section III-D, from the synthetic dataset, training and validation samples are offline sampled taking into consideration the intersections. The similarity network is composed of three fully connected layers with input, hidden and output dimensions of $d_i = 7$, $d_h = 32$ and

$d_o = 16$ neurons respectively. The similarity network is trained starting from randomly initialized weights for 50 epochs, with batch size 128, learning rate $5 \cdot 10^{-4}$, using Adam as optimizer and with the final weights selected based on the validation loss. As connection probability threshold t_c, the value of 0.2 is used throughout the experiments. As highlighted in Section III-E, this threshold comes into play only in case of not reliable masks M_b, i.e. false negatives and positives.

B. Baseline Methods

The DLO-specific approach named *Ariadne+* [10] is used as comparison. It employs the same segmentation network architecture of the one introduced in Section III-A to distinguish the DLOs from the scene. Thus, comparisons with *FASTDLO* can be established both at segmentation level, i.e. utilizing the weights of the original work [10] and the ones obtained from the synthetic dataset, and at the instance segmentation level, i.e. comparing the final result fixing the segmentation network and weights for both *Ariadne+* and *FASTDLO*. As described in [10], a number of superpixels equal to 50 is employed in *Ariadne+* for the comparisons.

The other baselines employed are general purpose DCNN performing the instance segmentation: *YOLACT* [14], *YOLACT++* [15], *BlendMask* [16] and *CondInst* [17]. Networks backbones with different depths can be applied in these DCNN models, thus comparisons are established for each configuration. The already introduced synthetic dataset of Section III-A, labeled in this case for the instance segmentation task, and with the mentioned train-val split $90 - 10$ is used in the training stage of each model. The hyper-parameters of each method have been tuned trying to maximize the performances. A general training strategy consisting of a maximum of 250 K iterations with the selection of the final weights based on the minimum validation loss has been followed for all the DCNN baselines. The augmentation schema resembles the one used for the semantic segmentation network. *YOLACT* and *YOLACT++* have been trained starting from the ImageNet weights with a batch size of 6, an initial learning rate of 10^{-3} reduced by a factor of 10 at iterations 100 K and 150 K. Stochastic gradient descent (SGD) with momentum 0.9 and weight decay $5 \cdot 10^{-4}$ is employed as optimizer. *BlendMask* and *CondInst* have also been trained starting from pre-trained weights on ImageNet with a batch size of 6 and with an initial learning rate of 0.01 reduced by a factor of 10 at iterations 100 K and 150 K. As optimizer SGD is employed with weights decay and momentum set to 10^{-4} and 0.9 respectively.

C. Test Dataset and Metrics

To evaluate the *FASTDLO* performances on real data, a *test set* of 135 manually labeled real images of electrical wires with varying diameters and collected in different real scenarios is used. The test dataset is organized into 3 categories, each containing 45 images:

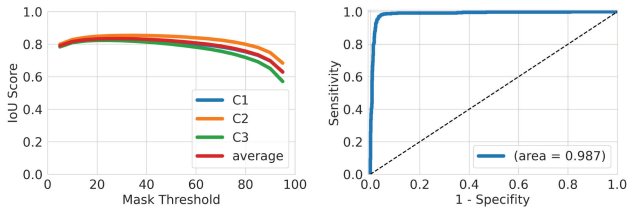
C1: scenes with the target wires laying on a surface and no other disturbing objects. The difficulties in these scenes are the high contrast shadows, possible chroma similarities with the background, the light settings and the perspective distortions.

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). The difficulties are given by the metallic surface reflecting the wires and other disturbing objects like commercial electromechanical components, typical of these products.

Each category is further divided in sub-classes based on the number of intersections present in the images, i.e. the subcategories 1 (one), 2 (two) and 3 (three) are created with 15 samples each. Compared to the test set employed in [10], here 45 new images have been added to evaluate the different diameters condition, 15 images for each category with 5 images in each subcategory. In the remainder of this section, the group of images corresponding to the *test set* of [10] is referenced as *base* while the new group of images as *ext*.

As a metric for the evaluation, 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. For the semantic segmentation network, the mask M corresponds to the binary mask M_b, while for the instance segmentation results 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.



(a) segmentation network IoU (b) similarity network ROC

Fig. 8. Networks performances in terms of IoU score and mask threshold in (a) for the semantic segmentation model, and receiver operating characteristic curve in (b) for the similarity-based neural network.

D. Evaluation

In Fig. 8 the plots related to the segmentation and similarity network performances on the *test set* are provided. For the first, i.e. Fig. 8(a), an almost constant IoU score is obtained for a wide range of masks' threshold values. Based on this plot, a value of 0.3 is selected as mask threshold. Concerning Fig. 8(b), the evaluation on each intersection in the *test set* images is performed by denoting a positive result if the predicted probability score between the correct endpoints is the largest among the complete set of scores obtained from the intersection under test. Thus, the Receiver Operating Characteristic (ROC) curve is built.

The baseline methods are evaluated in Table I by means of the IoU score computed starting from the color masks provided as output by each method. In addition, the table also provides

TABLE I
COMPARISON OF *FASTDLO* WITH BASELINES: *ARIADNE+* [10], *YOLACT* [14], *YOLACT++* [15], *BLENDMASK* [16] AND *CONDINST* [17]. RESNET-50 AND RESNET-101 ARE FROM [25]

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**

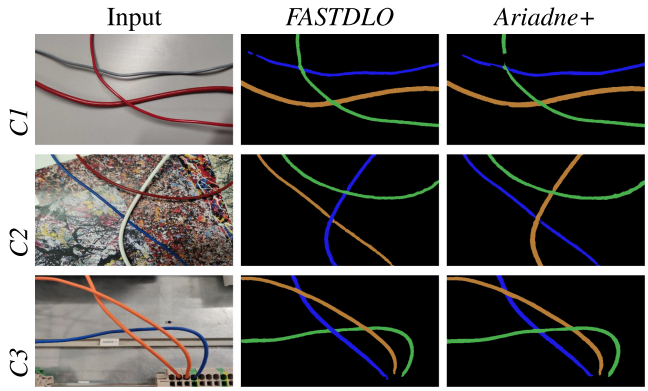


Fig. 9. Qualitative evaluation of *FASTDLO* and the best performing baseline using a sample for each category.

details about the average inference time and FPS of each method when applied to the *test set* plus a flag indicating if each approach provides as output an additional representation of the DLO instances in terms of key-points or splines, allowing for a broader comparison. *FASTDLO* achieves better overall scores showing a large advantage over the general purpose approaches and performing slightly better compared to *Ariadne+*, where both methods employ the same weights in the segmentation network. From the processing time perspective, *FASTDLO* is competitive with respect to the general purpose methods while being almost one order of magnitude faster than *Ariande+*. In Fig. 9 some samples for each *test set* category are shown with the corresponding output predictions obtained with *FASTDLO* and with *Ariande+*.

The prediction performances of the intersections layouts, i.e. Section III-F, are also evaluated on the *test set*. Considering only the correct endpoint-pairs predictions, the approach discussed in Section III-F is able to provide a correct result in 226 of the totals of 232 intersections achieving an overall accuracy of 97.4% compared to $78.3\%\,(177/226)$ of *Ariadne+*. Thus, it is clear the validity of the proposed method that is executed without noticeable overhead in terms of processing time.

E. Comparison Studies

The semantic segmentation performances on the *test set* are compared in Table II when deploying the synthetic dataset of Section III-A or the chroma-key dataset of [18] for the training. From the table it is observable that the average scores are very similar, with the synthetic dataset handling better the category *C1*, mostly due to the shadows, and the *ext* group of images consisting of cables with varying diameters. On the contrary, the dataset of [18] shows stronger performances on the complex industrial background category *C3*.

For a better evaluation of the *FASTDLO* performances with respect to *Ariadne+*, an *Hybrid* model is built starting from *FASTDLO* and replacing its intersection processing method (Section III-D) with the curvature and colors predictors of *Ariadne+*. The utility of the *Hybrid* model is thus twofold, since it allows to evaluate the benefits of: 1) the skeleton-based processing of the mask (Sections III-B and III-C); 2) the similarity-network (Section III-D). The results of the comparisons are shown in Table III. The skeleton-based processing of the masks allows to gain on average 0.50 in IoU score between *Ariadne+* and *Hybrid*. On the contrary, the similarity-network based processing of the intersections introduces an average gain of 0.35 in the IoU scores. The gains are mostly present in the *ext* group of the *test set* images, showing the limitation of *Ariadne+* in handling scenes with DLOs of different diameters.

TABLE II
COMPARISON OF THE SEMANTIC SEGMENTATION PERFORMANCES WHEN EMPLOYING THE SYNTHETIC DATASET OF SECTION III-A AND THE CHROMA-KEY DATASET [18]

Test	Synthetic			Chroma-key		
	base	ext	all	base	ext	all
C1	82.43	83.43	82.76	84.23	72.59	80.35
C2	84.04	89.31	85.80	85.17	89.01	86.45
C3	81.44	88.74	83.87	87.43	85.20	86.69
Avg	**82.76**	**87.16**	**84.14**	**85.61**	**82.27**	**84.49**

ResNet-101 is used as backbone. The values denote the IoU scores in percentage.

TABLE III
COMPARISON BETWEEN *FASTDLO*, *ARIADNE+* [10] AND THE *HYBRID* MODEL. THE VALUES DENOTE THE IOU SCORES IN PERCENTAGE

Train	Test	FASTDLO			Hybrid			Ariadne+ [10]		
		base	ext	all	base	ext	all	base	ext	all
Synthetic	**Avg**	**75.50**	**82.31**	**77.77**	**75.45**	**81.51**	**77.47**	**75.30**	**79.99**	**76.87**
Chroma-Key	**Avg**	**79.26**	**75.27**	**77.93**	**79.07**	**74.53**	**77.56**	**78.94**	**73.53**	**77.14**

TABLE IV
CHARACTERIZATION OF THE AVERAGE EXECUTION TIMES FOR THE MAIN STAGES OF *FASTDLO* WITH RESPECT TO THE NUMBER OF INTERSECTIONS IN THE IMAGE, I.E. 1, 2, AND 3, AND THE BACKBONE

Procedure	ResNet-101			ResNet-50		
	1	2	3	1	2	3
Binary Segmentation	19.72	19.49	19.49	15.73	15.67	15.67
Skeleton Generation	12.27	13.26	14.25	12.86	14.33	14.78
Endpoint-pairs Predictions	0.80	1.04	1.21	0.86	1.01	1.16
Informed Merging	15.13	17.09	18.81	15.93	18.57	19.58
Total	41.91	45.62	50.53	39.26	45.43	49.07

The values in the table are in milliseconds.

F. Timings

In Table IV a characterization of the average timing for the different stages of the method is provided by analyzing the effects of the ResNet backbones and the number of intersections in the image. The values are obtained by averaging the processing

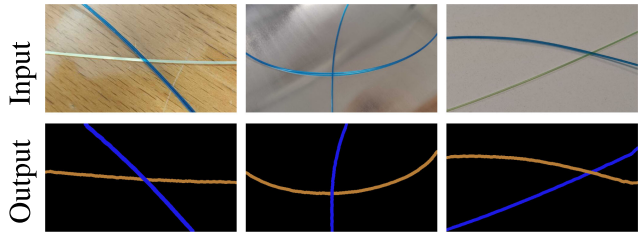


Fig. 10. *FASTDLO* applied to medical hoses.

time experienced on the *test set*. As the number of intersections increases, both the segmentation and endpoint-pairs predictions times stay relatively constant. The inference performed by the similarity network is indeed very fast and does not suffer significantly from the increase in the number of intersections to process thanks to the batch-inference. Instead, the skeleton generation time increases of about 15% from 1 to 3 intersections. Also, the additional processing time, mostly due to the informed merging approach, increases with the number of intersections, as expected. Overall, the total processing time is in the range of 40 to 50 ms in all the conditions.

G. Extensions to Other DLOs

FASTDLO can be easily extended to work with a large variety of DLOs. Fig. 10 displays the results of medical hoses segmentation where *FASTDLO* is applied directly without modifications. For other types of DLOs, like ropes and strings, where the texture characteristic of the surface of the objects can be different compared to the one of cables and wires, the semantic segmentation stage should be re-trained or fine-tuned. This can be accomplished easily by leveraging synthetic data, as shown in this paper. Apart from that, *FASTDLO* can also be directly applied to these objects.

V. Conclusions and Future Work

In this paper, a DLOs instance segmentation algorithm is presented featuring a processing rate higher than 20 FPS while preserving reliable and accurate predictions. The experimental results demonstrate the validity of *FASTDLO* when compared to several baselines available in the literature.

In future works, *FASTDLO* will be integrated into a robotic system for switchgears cabling. Moreover, the use of multiple camera frames will be investigated to improve the semantic segmentation of the scene. Finally, further refinements and optimization in the approach will be performed aiming towards real-time capabilities and, most importantly, a real-time tracking system for DLOs.

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," *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 688–716, 2018.

[2] X. Jiang, K.-m. Koo, K. Kikuchi, A. 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.

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

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

[5] A. Caporali, K. Galassi, and G. Palli, "3D DLO shape detection and grasp planning from multiple 2D views," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, 2021, pp. 424–429.

[6] A. Keipour, M. Bandari, and S. Schaal, "Deformable one-dimensional object detection for routing and manipulation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4329–4336, Apr. 2022.

[7] 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.

[8] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 479–484.

[9] 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 *Proc. Asian Conf. Comput. Vis.*. Springer, 2018, pp. 662–677.

[10] A. Caporali, R. Zanella, D. De Gregorio, and G. Palli, "Ariadne+: Deep learning-based augmented framework for the instance segmentation of wires," *IEEE Trans. Ind. Informat.*, doi: 10.1109/TII.2022.3154477.

[11] 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.

[12] M. Yu, H. Zhong, and X. Li, "Shape control of deformable linear objects with offline and online learning of local linear deformation models," 2021, *arXiv:2109.11091*.

[13] Y. Yang, J. A. Stork, and T. Stoyanov, "Learning to propagate interaction effects for modeling deformable linear objects dynamics," in *Proc. Int. Conf. Robot. Automat.*, 2021, pp. 1950–1957.

[14] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9157–9166.

[15] 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.

[16] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, "BlendMask: Top-down meets bottom-up for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8573–8581.

[17] Z. Tian, C. Shen, and H. Chen, "Conditional convolutions for instance segmentation," in *Proc. Eur. Conf. Comput. Vis.*. Springer, 2020, pp. 282–298.

[18] 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.

[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. Eur. Conf. Comput. Vis.*, 2016, pp. 909–916.

[21] A. Cirillo, G. De Maria, C. Natale, and S. Pirozzi, "Design and evaluation of tactile sensors for the estimation of grasped wire shape," in *Proc. IEEE Int. Conf. Adv. Intell. Mechatronics*, 2017, pp. 490–496.

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

[23] G. Borgefors, "Distance transformations in digital images," *Comput. Vis., Graph., Image Process.*, vol. 34, no. 3, pp. 344–371, 1986.

[24] 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.

[25] 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.