# mBEST: Realtime Deformable Linear Object Detection Through Minimal Bending Energy Skeleton Pixel Traversals

Andrew Choi, Dezhong Tong, Brian Park, Demetri Terzopoulos, *Life Fellow, IEEE*, Jungseock Joo, and Mohammad Khalid Jawed

***Abstract*—Robotic manipulation of deformable materials is a challenging task that often requires realtime visual feedback. This is especially true for deformable linear objects (DLOs) or "rods", whose slender and flexible structures make proper tracking and detection nontrivial. To address this challenge, we present *mBEST*, a robust algorithm for the realtime detection of DLOs that is capable of producing an ordered pixel sequence of each DLO's centerline along with segmentation masks. Our algorithm obtains a binary mask of the DLOs and then thins it to produce a skeleton pixel representation. After refining the skeleton to ensure topological correctness, the pixels are traversed to generate paths along each unique DLO. At the core of our algorithm, we postulate that intersections can be robustly handled by choosing the combination of paths that minimizes the cumulative bending energy of the DLO(s). We show that this simple and intuitive formulation outperforms the state-of-the-art methods for detecting DLOs with large numbers of sporadic crossings ranging from curvatures with high variance to nearly-parallel configurations. Furthermore, our method achieves a significant performance improvement of approximately 50% faster runtime and better scaling over the state of the art.**

***Index Terms*—Deformable linear objects, DLOs, instance segmentation, computer vision, perception for manipulation.**

## I. Introduction

As ROBOTS become increasingly more intelligent and capable, developing robust and effective deformable material manipulation skills has started to attract substantial research attention [1]. Among various deformable objects, deformable linear objects (DLOs) — typically referred to as "rods" by the mechanics community — are a special group, including everyday objects such as cables, ropes, tubes, and threads. Due to their distinctive geometric characteristic (width $\approx$ height $\ll$ length), DLOs are widely used in various domestic and industrial applications, including surgical suturing [2], knot fastening [3], [4], cable manipulation [5], [6], food manipulation [7], mechanics analysis [8], and more. Because of their flexibility, DLOs are often prone to complex tangling, which complicates manipulation. Additionally, the complicated structures made by DLOs usually have unique topology-induced mechanical properties [9], [10], [11], [12], [13] and are, therefore, used to tie knots for sailing, fishing, climbing, and various other engineering applications. Given all the aforementioned, a robust, efficient, and accurate perception algorithm for DLOs is crucial to both deformable material manipulation and soft robotics.

We present an algorithm for robust, accurate, and fast instance segmentation of DLOs, named *mBEST* (Minimal Bending Energy Skeleton pixel Traversals). Without any prior knowledge regarding the geometries, colors, and total number of DLOs, *mBEST* takes a raw RGB image as input and outputs a series of ordered pixels defining the centerline of each individual DLO in the image, thus allowing for the configurations of different DLOs to be easily incorporated into motion planning and manipulation schemes.

To achieve instance segmentation of DLOs in images, we implement the following sequence of processing steps: Like previous work [14], we first perform semantic segmentation to produce a binary mask of the DLOs against the background using either simple color filtering methods or a Deep Convolutional Neural Network (DCNN). After a binary mask is obtained, we apply a thinning algorithm to the mask to produce a single-pixel-wide skeleton representation of the DLOs, which preserves the connectivity and centerlines of the binary mask. Thus, key points such as ends and intersections are easily detected. After a series of refinement steps to ensure topological correctness, the skeleton is then traversed, one end at a time, in a manner that minimizes the cumulative bending energy of the DLOs, until another end is encountered. Each traversal yields a single DLO's centerline pixel coordinates, which optionally can then be used to produce segmentation masks. Fig. 1 overviews the *mBEST* processing pipeline.

Overall, our main contributions in this article are that we

1) develop a robust pipeline for obtaining ordered centerline coordinates and segmentation masks of DLOs from semantic binary masks;

Manuscript received 18 February 2023; accepted 12 June 2023. Date of publication 28 June 2023; date of current version 5 July 2023. 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 National Science Foundation under Awards OAC-2209782, CMMI-2101751, CAREER-2047663, and IIS-1925360. *(Corresponding author: Mohammad Khalid Jawed.)*

Andrew Choi, Brian Park, and Demetri Terzopoulos are with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA (e-mail: asjchoi@cs.ucla.edu; briannparkk@ucla.edu; dt@cs.ucla.edu).

Dezhong Tong and Mohammad Khalid Jawed are with the Department of Mechanical and Aerospace Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: tltl960308@g.ucla.edu; khalidjm@seas.ucla.edu).

Jungseock Joo is with the Department of Communication, University of California, Los Angeles, CA 90095 USA, and also with NVIDIA Corporation, Santa Clara, CA 95051 USA (e-mail: jjoo@comm.ucla.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3290419, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3290419

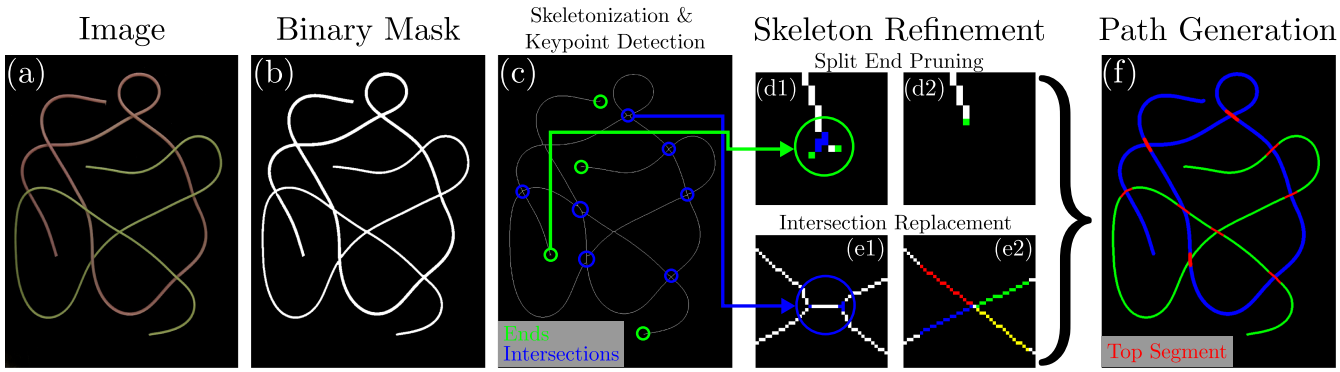


Fig. 1. Overview of the *mBEST* processing pipeline. An input image (a) is converted to a binary mask (b) using a segmentation method. The binary mask is then converted to a skeleton pixel representation (c), where the connectivity and centerlines of the DLOs are preserved as a single-pixel wide structure and keypoints, such as intersections and ends, are detected. This is followed by a series of refinement steps to maintain the topological correctness of the skeleton: Split ends (d1) are pruned (d2) and pixels representing a single topological intersection (e1) are clustered, matched, and replaced with a more intuitive intersection (e2). Finally, the DLOs are delineated (f) by traversing skeleton pixels and choosing minimal cumulative bending energy paths.

2) demonstrate that the relatively simple and physically meaningful optimization objective of minimizing cumulative bending energy outperforms several state of the art (SOTA) algorithms;
3) showcase the effectiveness of our topology-correcting skeleton refinement steps by outperforming the SOTA algorithms with a hybrid *mBEST* formulation that uses the intersection handling scheme of SOTA algorithms;
4) achieve faster, real-time performance compared to the SOTA algorithms.

Moreover, we have released all our source code, datasets (with ground truth), and a supplementary video.[1]

The remainder of the article is organized as follows: We present a review of related work in Section II. The algorithmic formulation of *mBEST* is then detailed in Section III. In Section IV, we report our experimental results comparing *mBEST* with the SOTA approaches. Finally, we make concluding remarks and discuss potential future research directions in Section V.

## II. Related Work

Although research into manipulation skills for DLOs has been prevalent, the perception algorithms used in support of these efforts remain underdeveloped. For example, in the work of Tong et al. [8], attached markers are required to determine the configuration of the manipulated DLO. Zhu et al. [5] carefully adjusted the workspace to increase the contrast between the manipulated DLOs (cables) and their background. Although these prior efforts successfully completed their target manipulation tasks, the simplistic perception algorithms restrict real world applicability.

Consequently, DLO detection algorithms featuring various methodologies have been proposed. Keipour et al. [15] evaluated both curvatures and distances to fit a continuous DLO. Using data-driven methods, Yan et al. [16] trained a neural network to reconstruct the topology of a DLO based on a coarse-to-fine nodal representation. Though these methods achieve good results for some datasets, they work under the strict assumption that only one DLO exists within the scene, which dramatically restricts their applicability.

One of the first perception algorithms capable of detecting multiple DLOs, *Ariadne* [17], segments images into superpixels and traverses the superpixels belonging to DLOs in order to produce paths. The ambiguity of intersections is handled using a multi-faceted cost function that takes into consideration color, distance, and curvature. Despite its satisfactory performance, this early approach suffers from a large number of hyperparameters, an overreliance on DLOs being a uniform color, and the tedious requirement that the user manually select the ends of DLOs. Furthermore, the processing speed of *Ariadne* is on the order of seconds, precluding realtime operation.

In recent years, data-driven computer vision methods have attracted increasing attention and researchers have shown that image segmentation problems can be tackled efficiently and accurately using Deep Convolutional Neural Networks (DCNNs), particularly instance segmentation [18], [19], [20], [21]. Furthermore, techniques have been introduced to help synthetically generate large quantities of photorealistic data in order to adequately train such models [22], [23], [24]. Using DCNNs, Zanella et al. [25] created segmentations of DLOs such as wires; however, the segmentations did not distinguish between each DLO.

Improving upon *Ariadne*, *Ariadne+* [26] also utilizes a DCNN model to extract an initial binary mask of the DLOs. This allows the algorithm to then apply superpixel segmentation purely on the binary mask itself, significantly reducing the computation time. Paths are then generated in a similar fashion to the original *Ariadne* algorithm by traversing superpixels while intersections are handled using a neural network to predict the most probable paths. Despite these improvements, *Ariadne+* is sub-realtime; i.e., less than 3 FPS.

Another algorithm, *FASTDLO* [14] improves upon the speed of *Ariadne+* by forgoing superpixel segmentation altogether. Instead, it uses a skeleton pixel representation of the DLO binary mask for path traversals. Intersections are then also handled by a neural network. By replacing superpixel segmentation with skeletonization, *FASTDLO* is able to achieve a realtime performance of 20 FPS for images of size $640 \times 360$ pixels.

More recently, *RT-DLO* [27] detects DLOs by representing them as sparse graphs where nodes are sampled from DLO

[1] See https://github.com/StructuresComp/mBEST.

TABLE I
COMPARISON OF ALGORITHMS

| Algorithm | Intersection Rule | DLO Representation | Realtime |
|---|---|---|---|
| Ariadne [17] | color, distance, curvature | superpixels | ✗ |
| Ariadne+ [26] | DNN prediction | superpixels | ✓⁻ |
| FASTDLO [14] | DNN prediction | skeleton pixels | ✓ |
| RT-DLO [27] | cosine similarity | sparse graph | ✓ |
| mBEST | curvature | skeleton pixels | ✓ |

centerlines and edges are selected based on topological reasoning. This results in increased runtime efficiency and accuracy compared to *Ariadne+* and *FASTDLO*, but requires sampling along the centerlines of the DLO to remain computationally competitive, often resulting in noisy segmentations. Furthermore, several hyperparameters must be set.

*Ariadne+*, *FASTDLO*, and *RT-DLO* are considered state-of-the-art DLO perception algorithms, but they have been evaluated only on scenes containing DLOs with relatively smooth curvatures and minimal self-loops. Our experiments will show that these algorithms struggle to resolve nontrivial configurations (e.g., DLOs with highly variable curvatures resulting in many crossings and tangles and/or nearly-parallel intersections). We argue that a physically principled approach can outperform both sparse graphs and black box neural network approaches when dealing with intersections. Our *mBEST* algorithm robustly solves complex scenes using the simple notion that the most probable path is the one that minimizes cumulative bending energy. Not only does *mBEST* outperform in accuracy, it also achieves realtime performance with a 50% improvement over the next best algorithm and it has no hyperparameters to set. Table I summarizes the key algorithmic differences between *mBEST* and competing algorithms.

## III. METHODOLOGY

The *mBEST* algorithm consists of the following steps:
1) DLO Segmentation
2) Skeletonization
3) Keypoint Detection
4) Split End Pruning
5) Intersection Clustering, Matching, and Replacement
6) Minimal Bending Energy Path Generation
7) Crossing Order Determination

The following sections describe each step in detail.

### A. DLO Segmentation

The first step in detecting the DLOs is to obtain a binary mask $\mathbf{M}_{\text{dlo}}$ of the image that distinguishes all DLO-related pixels from the background. The initial image segmentation method is not a key contribution of *mBEST*. Rather, it is a modular component of our pipeline, allowing for different methods to be plugged in depending on the use case. As stated previously, we employ two semantic segmentation methods: a DCNN segmentation model and color filtering. In particular, we use *FASTDLO*'s pretrained DCNN model [14] in our experiments.

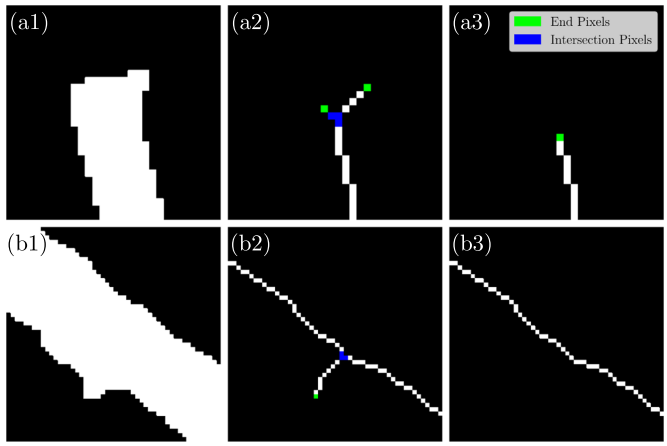


Fig. 2. Examples of split ends that may occur during the skeletonization process. Row (a) shows split ends that may occur at an actual topological end, while row (b) shows a split end along a segment produced by a jagged mask. For both examples, the first column shows the binary mask; the second shows the split end after skeletonization, and the third shows the topologically correct structure after pruning.

### B. Skeletonization

As shown in Fig. 1(b)–(c), the next step of our algorithm is to convert $\mathbf{M}_{\text{dlo}}$ to a skeleton mask $\mathbf{M}_{\text{sk}}$, which is useful as both the connectivity and general topology of the DLOs are maintained. Furthermore, as segments are only 1 pixel wide, traversals along segments are not susceptible to path ambiguity. To achieve skeletonization, we use an efficient thinning algorithm designed specifically for 2D images, and refer the reader to [28] for the details.

### C. Keypoint Detection

After obtaining the skeleton pixel representation, we can then detect two types of key points: ends and intersections. Locating ends is crucial since they serve as the start and finish points for skeleton pixel traversals. Locating intersections is crucial as they represent the only points at which a pixel traversal will have multiple possible routes. Therefore, care must be taken in choosing the correct path when passing through an intersection.

To detect ends and intersections, a skeleton pixel classification kernel,

$$\mathbf{K} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 10 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

is convolved with the skeleton mask; i.e., $\mathbf{M}_{\text{sk}} \circledast \mathbf{K}$. We then identify all end pixels $\mathbf{E}$ as those with a value of 11 (1 neighbor) and all intersection pixels $\mathbf{I}$ as those with a value greater than 12 (3 or more neighbors).

After obtaining both $\mathbf{E}$ and $\mathbf{I}$, additional work must be done to obtain the correct representative sets. For example, end pixels that are unindicative of a topological end may be produced from a noisy binary mask. These "split ends" will then falsely produce intersection pixels themselves, as shown in Fig. 2. Additionally, a single topological intersection will result in either two Y-shaped divides or a single X-shaped divide, as shown in Fig. 3(a). Such pixels must be clustered accordingly, with a single point of intersection determined. In the case of a skeleton possessing

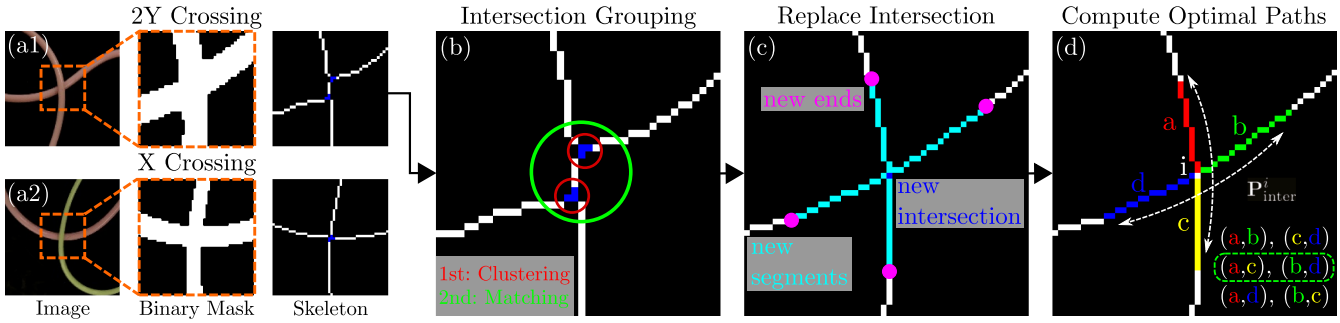


Fig. 3. Intersection clustering, matching, replacement, and optimal path generation pipeline. Two sample intersections are shown where skeletonization results in a 2Y-shaped crossing (a1) and an X-shaped crossing (a2). As 2Y-shaped crossings are topologically incorrect, we replace them by replacing the intersection pixels (b) in two stages: The first involves clustering adjacent pixels and the second involves pair matching nearby clusters. Using the centroid location of the matched clusters, we then replace the intersection (c) by creating new ends and having new segments sprout and connect to the centroid. Finally, (d) the new generated ends and segments are used to discover the combination of paths that minimizes the cumulative bending energy of the DLO.

two Y-shaped divides in the context of a single intersection, the intersection must also be replaced with an X-shaped divide that more accurately represents the centerlines of the DLOs.

### D. Split End Pruning

When the boundary of the binary mask $\mathbf{M}_{\text{dlo}}$ is jagged, the skeleton mask $\mathbf{M}_{\text{sk}}$ may contain several types of split ends, as shown in Fig. 2. Such split ends must be identified and pruned as they do not accurately represent the topology of the DLO(s) and will result in incorrect start points as well as cause path ambiguity during pixel traversals.

Note that the length of a split end can be at most the radius of the DLO from which it is sprouting. Therefore, the length of all split ends should be within a threshold $\delta$ much less than the length of the DLO. As such, for every end in $\mathbf{E}$, we traverse along its segment until one of the following three conditions occurs before traversing $\delta$ pixels:

1) an intersection is encountered,
2) an end is encountered,
3) or neither was encountered.

For Conditions 1 and 2, we remove the segment that was just traversed from $\mathbf{M}_{\text{sk}}$ as well as the corresponding end from $\mathbf{E}$. For Condition 1, we must also remove from $\mathbf{I}$ all intersection pixels that were produced from the pruned split end. For any endpoint that satisfies Condition 3, we do nothing.

To encompass all possible split ends, we can set $\delta$ to be the diameter of the widest DLO in the image. Radii of the DLOs can be obtained by computing an L2 distance transform on $\mathbf{M}_{\text{dlo}}$, which results in a matrix $\mathbf{D}$ containing for each pixel location the closest Euclidean distance to a 0-value pixel. With this, we can then simply set $\delta = 2\max(\mathbf{D})$. As the distance matrix $\mathbf{D}$ tells us the radii information for all centerline points, we can reuse it to generate segmentation masks once each DLO's path is ascertained [14].

### E. Intersection Clustering, Matching, and Replacement

As mentioned in Section III-C, a single topological intersection can result in either a 2Y or X-shaped branching as shown in Fig. 3(a). Furthermore, each of these branches may have several intersection pixels; i.e., pixels with 3 or more neighbors. Our goal then is to group each pixel in $\mathbf{I}$ to a single branch and then group each branch to its true topological intersection. With all the intersection pixels properly grouped, we then define a single intersection pixel that represents the true center of a crossing, for all crossings.

First, to cluster all adjacent intersection pixels, we use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [29], an algorithm that clusters data points within a distance threshold of each other. In our case, the Euclidean pixel distance and is simply set to 2. Once all adjacent pixels in $\mathbf{I}$ are clustered, each cluster is averaged to create a new $\mathbf{I}$.

The next step is to group all branches in $\mathbf{I}$ by their respective topological intersection. To do so, we first classify all branches in $\mathbf{I}$ as either Y or X-branches. Intersections that are X-branches are already topologically correct so they are left unmodified in $\mathbf{I}$. The remaining Y-branches are removed from $\mathbf{I}$, after which we obtain a list of all possible Y-branch pair combinations and sort them by their pair distance. The closest branch pairs are then iteratively popped from the list and matched so long as neither branch has already been matched. A new intersection pixel is then computed from the average of the matched branch locations and then added back to $\mathbf{I}$.

Using the new intersection pixel location, all matched Y-branches can then be replaced with an X-shaped branch, as shown in Fig. 3(c). Note that there may be cases where an intersection is topologically a Y-branch (i.e., an end perfectly overlaps with a segment) and thus has no corresponding match. To account for these cases, we stop matching Y-branches once the pair distance exceeds a limit $\epsilon = 10\max(\mathbf{D})$ or if every Y-branch has already been matched. Any remaining non-matched Y-branches are added back to $\mathbf{I}$. As shown in Fig. 3(c), we record new "ends" for all topologically correct intersections. This is done so that we know that an intersection is imminent during a pixel traversal and, hence, take the correct precomputed path, as discussed next.

### F. Minimal Bending Energy Path Generation

For rods that have nonuniform curvatures, the bending energy must be computed in a discretized fashion. If we discretize a rod into $N$ nodes and $N-1$ edges, then the total bending energy is

$$E_b = \frac{1}{2}\frac{EI}{V}\sum_{k=1}^{N-2}\left(\kappa_k - \kappa_k^0\right)^2, \tag{1}$$

where $EI$ is the bending stiffness, $\kappa_k$ and $\kappa_k^0$ are the deformed and undeformed discrete dimensionless curvatures, respectively, at node $k \in [1, N-2]$, and $V$ is the Voronoi length. For our DLOs, we assume that the undeformed curvature is a straight configuration ($\kappa^0 = 0$). Then, minimizing the bending energy of an elastic rod amounts to minimizing the discrete curvatures.

The norm of the discrete dimensionless curvature for a node $k$ is easily computed using the unit tangent vectors of the adjacent edges [30]:

$$\bar{\kappa}_k = \left\| \frac{2\mathbf{t}^{k-1} \times \mathbf{t}^k}{1 + \mathbf{t}^{k-1} \cdot \mathbf{t}^k} \right\|, \tag{2}$$

where $\mathbf{t}^{k-1}$ and $\mathbf{t}^k$ are the unit tangent vectors of edges $k-1$ and $k$, respectively.

Note that the only time we must choose between multiple paths is at an intersection, whereas traversals through segments are unambiguous. Using the new ends shown in Fig. 3(c), we can compute the combination of paths that minimizes the cumulative bending energy of the DLOs by simply computing the pairs of segments that minimize cumulative norm curvature. In other words, if an intersection at $\mathbf{i}$ has four end points $\mathbf{a}, \mathbf{b}, \mathbf{c}$, and $\mathbf{d}$, then we must find the pairs of end points $(\mathbf{p}_1^1, \mathbf{p}_1^2)$ and $(\mathbf{p}_2^1, \mathbf{p}_2^2)$ that minimizes $\|\kappa_1\| + \|\kappa_2\|$, where

$$\begin{aligned}
\kappa_1 &= \frac{2\mathbf{t}_1^1 \times \mathbf{t}_1^2}{1 + \mathbf{t}_1^1 \cdot \mathbf{t}_1^2}, \quad \kappa_2 = \frac{2\mathbf{t}_2^1 \times \mathbf{t}_2^2}{1 + \mathbf{t}_2^1 \cdot \mathbf{t}_2^2}, \\
\mathbf{t}_1^1 &= \frac{\mathbf{i} - \mathbf{p}_1^1}{\|\mathbf{i} - \mathbf{p}_1^1\|}, \quad \mathbf{t}_1^2 = \frac{\mathbf{p}_1^2 - \mathbf{i}}{\|\mathbf{p}_1^2 - \mathbf{i}\|}, \\
\mathbf{t}_2^1 &= \frac{\mathbf{i} - \mathbf{p}_2^1}{\|\mathbf{i} - \mathbf{p}_2^1\|}, \quad \mathbf{t}_2^2 = \frac{\mathbf{p}_2^2 - \mathbf{i}}{\|\mathbf{p}_2^2 - \mathbf{i}\|}.
\end{aligned} \tag{3}$$

Fig. 3(d) shows an example of this optimization, where out of the 3 possible combinations of paths the one that minimizes total curvature is selected. With the paths through intersections properly precomputed, the skeleton pixel traversals to obtain each DLO's centerline can now take place. Algorithm 1 shows the pseudocode of the *mBEST* pipeline.

**Algorithm 1:** *mBEST* Pipeline Pseudocode.

```
Input: M_dlo
Output: P
Func mBEST(M_dlo):
    P ← [ ]
    M_sk ← Skeletonize(M_dlo)
    D ← DistTransform(M_dlo)
    δ, ϵ ← ComputeParams(D)
    E, I ← DetectKeyPoints(M_sk ⊛ K)
    E, I ← PruneSplitEnds(E, I, M_sk, δ)
    I ← ReplaceIntersections(I, ϵ)
    P_inter ← GenIntersectionPaths(I, M_sk)
    while E is not empty do
        x ← E.pop()
        while True do
            τ ← traverse along M_sk from x until reaching
              an end e
            if e ∈ P_inter then
                τ ← τ + P^i_inter
                x ← last pixel of P^i_inter
            else
                E.remove(e)
                break
        P.append(τ)
    return P
```

### G. Crossing Order Determination

The final step of the pipeline involves ascertaining which DLO is resting on top at intersections. To solve this problem, we use a modified version of *FASTDLO*'s [14] solution. To compute crossing order at intersections, we use the precomputed optimal paths shown in Fig. 3(d). Crossing order is then determined by computing the sum of the standard deviations of the RGB channels of the pixels along each path. Finally, the path that contains the lower sum is assumed to be the one on top. Although this solution from *FASTDLO* works fairly well, we discovered that failures can occur due to glare along the centerline, which may even cause failures for intersections with two completely different colored DLOs. To eliminate the influence of glare, we compute the standard deviations of the intersection path pixels not on the original input image but on its blurred version.

## IV. Experimental Results

### A. Datasets

We used two different datasets to evaluate the effectiveness of *mBEST*. The first consists of relatively simple configurations of DLOs against complex backgrounds, whereas the second consists of complex configurations (i.e., highly varying curvatures and numerous self-loops) of DLOs against a simple black background. We focused mostly on images with a simple black background since the initial binary mask segmentation is not a key aspect of our algorithm; however, *mBEST* also works well for complex backgrounds, as shown in Fig. 4.

The complex background dataset was provided by [27], and comprises a total of 132 images of size $640 \times 360$. It is split into tiers C1, C2, and C3, each containing 44 images, where the tier numbers reflect the increasing complexity of the background. Given the complexity of the background, DCNN segmentation was used to obtain the initial binary mask. We removed two images each from C2 and C3 as they included intersections involving $> 2$ DLOs, scenarios which are currently outside the scope of *mBEST*.

The simple background dataset consists of a total of 300 images of size $896 \times 672$ and is split into tiers S1, S2, and S3, each containing 100 images, where the tier numbers reflect the number of DLOs in the image, resulting in both an increase in complexity and computational demand as the numbers increase. Given the high contrast background, color filtering sufficed to obtain the initial binary mask.

### B. Baselines and Parameters

We tested *mBEST* against three state-of-the-art baselines: *Ariadne+* [26], *FASTDLO* [14], and *RT-DLO* [27]. In terms of hyperparameters, the number of superpixels for *Ariadne+* was set to 75 for complex background images and to 200 for simple background images. Both these values were chosen as optimal after performing a parameter sweep on each dataset. For *RT-DLO*, the K-nearest neighbors matching parameter was set to 8, the edge similarity threshold was set to 0.1, and the vertex

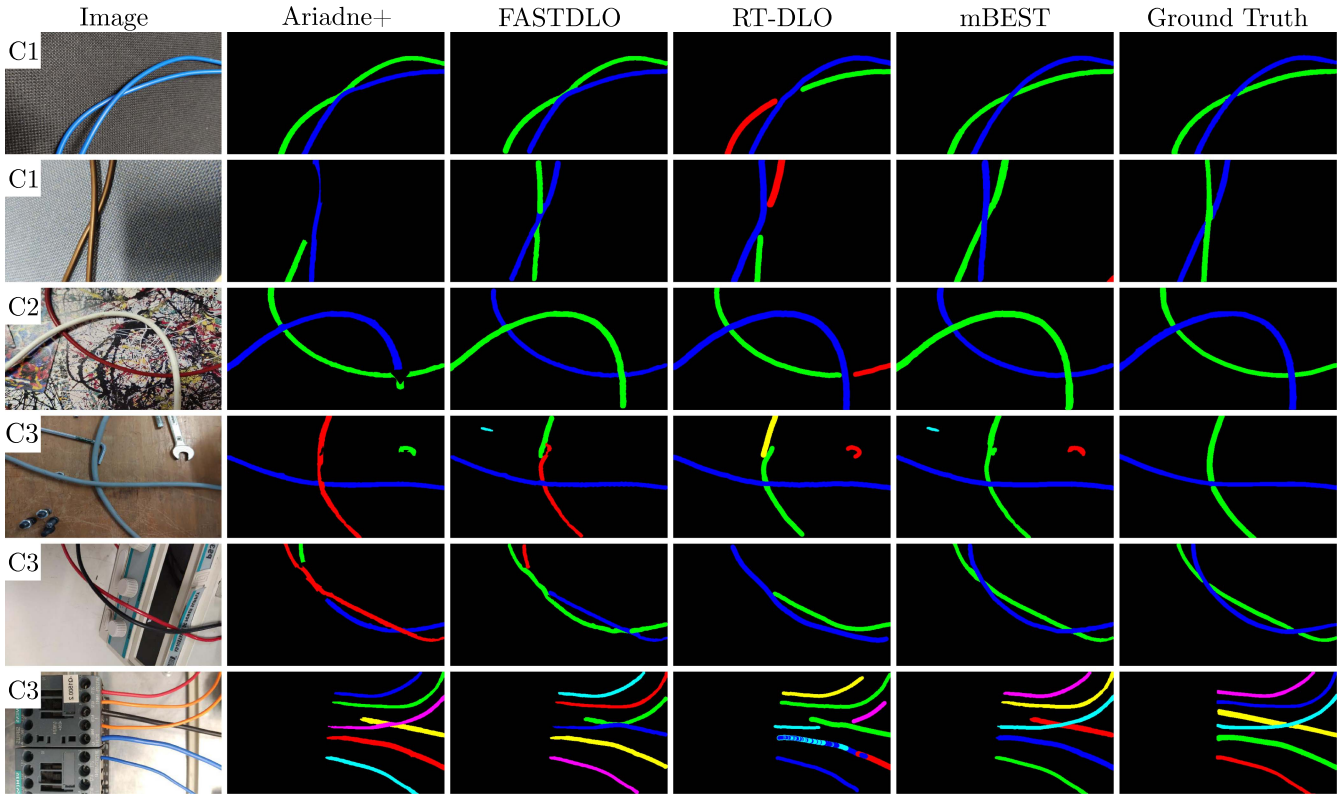


Fig. 4. Sample segmentations for the simple configuration against complex background dataset. Each row shows segmentation results for a different image with the left column indicating the dataset to which the image belongs. Columns 2–5 show *Ariadne+*, *FASTDLO*, *RT-DLO*, and *mBEST* results, respectively. The right column shows the ground truth. Note the failure to properly handle intersections for all baseline algorithms, especially when strands are nearly parallel. In fact, *RT-DLO* can be seen to produce an unintuitive output for the last example where certain wires are labeled multiple times.

TABLE II
EXPERIMENTAL RESULTS

| Dataset | DICE [%] | | | | Runtime [FPS] | | | |
|---|---|---|---|---|---|---|---|---|
| | Ariadne+ | FASTDLO | RT-DLO | mBEST | Ariadne+ | FASTDLO | RT-DLO | mBEST |
| C1 | $88.30 \pm 0.102$ | $89.82 \pm 0.091$ | $90.31 \pm 0.085$ | $\mathbf{91.08 \pm 0.083}$ | 2.69 | 20.81 | 30.58 | **31.86** |
| C2 | $91.03 \pm 0.044$ | $91.45 \pm 0.039$ | $91.10 \pm 0.058$ | $\mathbf{92.17 \pm 0.050}$ | 2.63 | 20.90 | **32.50** | 32.03 |
| C3 | $86.13 \pm 0.123$ | $86.55 \pm 0.110$ | $87.27 \pm 0.128$ | $\mathbf{89.69 \pm 0.089}$ | 2.72 | 20.51 | **32.44** | 32.17 |
| S1 | $97.24 \pm 0.065$ | $87.91 \pm 0.062$ | $96.72 \pm 0.014$ | $\mathbf{98.21 \pm 0.006}$ | 0.92 | 21.88 | 39.60 | **52.79** |
| S2 | $96.81 \pm 0.074$ | $88.92 \pm 0.061$ | $94.91 \pm 0.019$ | $\mathbf{97.10 \pm 0.010}$ | 0.78 | 17.34 | 25.73 | **41.04** |
| S3 | $96.28 \pm 0.067$ | $90.24 \pm 0.042$ | $94.12 \pm 0.043$ | $\mathbf{96.98 \pm 0.009}$ | 0.73 | 15.33 | 22.06 | **37.11** |

sampling ratio was set to 0.15. These hyperparameters were provided by default and shown to have good performance in [27]. For all experiments involving use of the DCNN model, a pixel segmentation threshold of 77 (0–255) was used. Furthermore, although *Ariadne+* has its own neural network for the initial segmentation of the DLOs, we replaced it with *FASTDLO*'s DCNN model for consistency and better performance.

Additionally, we demonstrated the effectiveness of *mBEST*'s skeleton refinement steps by conducting experiments on an aggregated dataset consisting of S1, S2, and S3 with a hybrid formulation that uses *FASTDLO*'s intersection handling neural network in *mBEST*'s framework.

All experiments were run on a workstation with an Intel i9-9900KF CPU and an NVIDIA RTX 2080 Ti GPU.

### C. Results and Analysis

We report two key metrics. First, we look at segmentation accuracy using the popular DICE metric. We also report the average run times for each algorithm in frames per second (FPS). Table II reports both metrics for all our experiments.

For the complex background datasets, we see that *mBEST* outperforms all baseline algorithms in terms of mean DICE score. In particular, we see that the baseline algorithms often struggle to handle intersections that are nearly parallel, as shown in Fig. 4.

With regard to runtime, *mBEST* is roughly on par with *RT-DLO* and is a clear improvement over *Ariadne+* ($\approx 11\times$) and *FASTDLO* ($\approx 1.5\times$). Note three important caveats for these results: 1) the initial DCNN segmentation comprises a

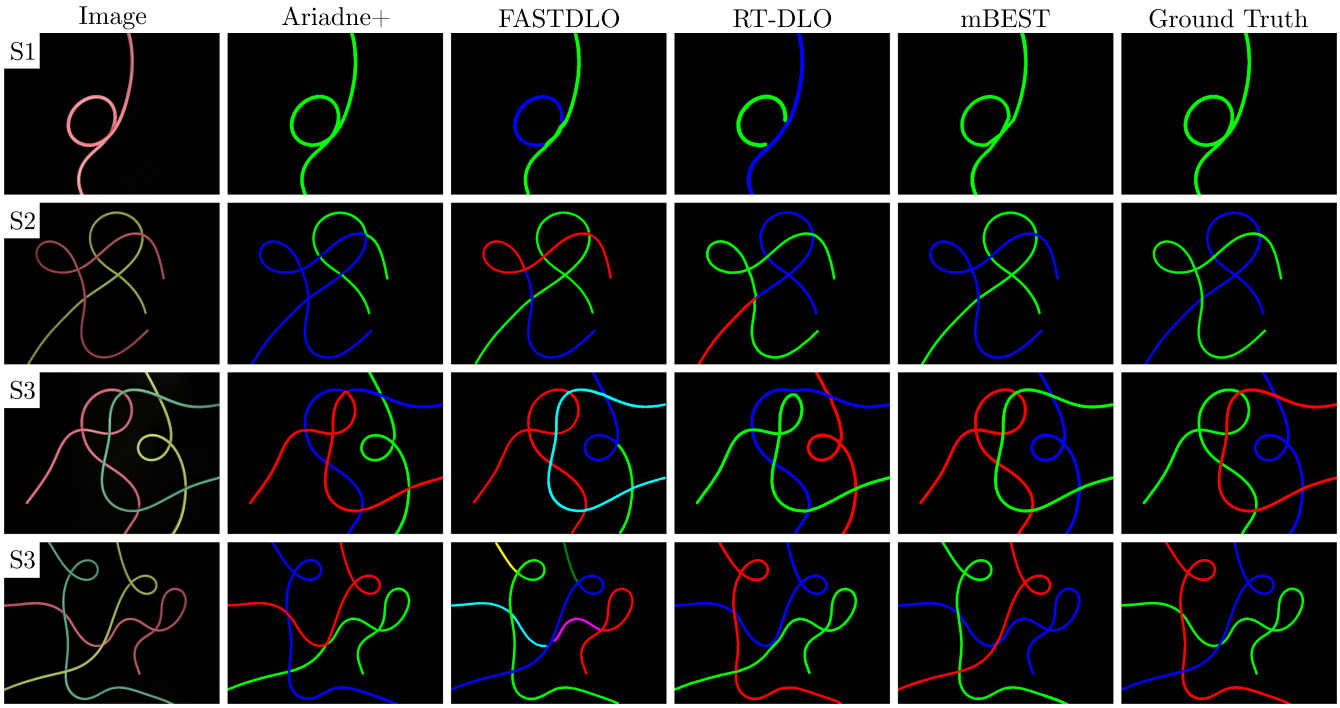


Fig. 5. Sample segmentations for the complex configuration against simple background dataset. Each row shows segmentation results for a different image with the left column indicating the dataset to which the image belongs. Columns 2–5 show results for *Ariadne+*, *FASTDLO*, *RT-DLO*, and *mBEST*, respectively. The right column shows the ground truth. Several cases of incorrect intersection handling can be observed for all the baseline algorithms, whereas *mBEST* robustly handles intersections using its simple bending energy optimization.

large portion of the computation time; 2) the images are relatively small and the number $N$ of DLOs is random, giving little insight as to how the algorithms scale with $N$, and 3) *RT-DLO*'s ability to keep up with *mBEST* in speed is solely due its low vertex sampling rate (0.15). We observe that increasing the sampling rate increases the compute time significantly given the computational expense of graph construction.

To address the above concerns, consider the results for the simple background datasets. As these datasets do not require the use of a DCNN and are labeled by the number of DLOs they contain, we can accurately determine how each algorithm scales and performs with respect to $N$. As reported in the bottom half of Table II, *mBEST* offers clear speed improvements over the *Ariadne+* ($\approx 54\times$), *FASTDLO* ($\approx 2.4\times$), and *RT-DLO* ($\approx 1.5\times$) baselines. Additionally, we see that *mBEST* scales better with respect to $N$ compared to *RT-DLO* despite the latter's sparse sampling rate, with *mBEST* experiencing runtime decreases of about 22.3% and 9.6% when moving up each tier compared to *RT-DLO*'s 35% and 14.3%. Though a low sampling rate works well for the relatively straight configurations of DLOs in C1, C2, and C3, we notice that performance degrades significantly once a coarse sampling rate is unable to capture the highly variable curvatures of complex assemblies of rods (i.e., those in S1, S2, and S3). Examples of this can be observed in our supplementary video (see Footnote 1).

In addition to the significant improvement in runtime, *mBEST* also outperforms all the baseline algorithms in terms of mean DICE score as well. Several examples of intersection failures experienced by the baseline algorithms are shown in Fig. 5. Such failures typically occur in extreme cases (i.e., either nearly-parallel or extremely curved self-loops). Interestingly, *Ariadne+*'s mean DICE score is very close to *mBEST*'s, but had up to $10\times$ the standard deviation, meaning that *Ariadne+* suffered a higher number of outright failures. In fact, *mBEST* has a lower standard deviation compared to all the baseline algorithms across all the datasets with the exception of C2, indicating a higher level of consistency for a wide range of data.

TABLE III
SKELETON REFINEMENT ANALYSIS ON S1+S2+S3

| Algorithm | DICE [%] | Runtime [FPS] |
|---|---|---|
| FASTDLO | $89.02 \pm 0.056$ | 16.13 |
| HYmBEST | $97.39 \pm 0.013$ | 29.94 |
| mBEST | $97.43 \pm 0.010$ | 42.86 |

Finally, we analyze the effectiveness of our skeleton refinement steps by formulating a hybrid algorithm, *HYmBEST*, which uses *FASTDLO*'s intersection handling neural network (IHNN) plugged into *mBEST*'s framework. As reported in Table III, *HYmBEST* achieves a mean DICE score almost identical to *mBEST*, with both significantly outperforming *FASTDLO*. This is noteworthy as it shows that *FASTDLO*'s IHNN works reasonably well, but that the improperly handled skeleton structure yields poor results, thus highlighting the importance of the topology-correcting refinement steps. Note also that although *FASTDLO*'s IHNN can perform well in a hybrid formulation, *mBEST*'s remarkably cheap bending energy formulation still results in an $\approx 43\%$ runtime improvement.

## V. Conclusion

We have introduced *mBEST*, an end-to-end pipeline for the segmentation of deformable linear objects (DLOs) in images that improves upon the state of the art both in terms of accuracy and computational speed. Through a variety of experiments, we have shown that *mBEST* can robustly handle complex scenes with highly tangled DLOs by generating paths on topologically correct skeletons that minimize the cumulative bending energy of the scene.

In future work, we will explore solutions that take into consideration occlusions, multiple DLOs at an intersection, poor quality binary masks, and dense knots; i.e., strands touching in parallel. We note that though we do not cover it in this manuscript, the bending energy formulation of *mBEST* can easily be expanded to deal with multiple DLOs at an intersection by simply accounting for additional path combinations. Furthermore, methods like *RT-DLO* [27] already take into consideration the possibility of poor binary masks and may be better suited for such situations. Finally, another promising research direction is 3D detection of DLOs, thus enabling robots to go beyond simple planar manipulation. Solutions for this may involve using *mBEST* to generate segmentations from multiple viewing angles for the purposes of 3D reconstruction.

## 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] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4111–4117.

[3] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery that learns to tie knots using recurrent neural networks," *Adv. Robot.*, vol. 22, no. 13/14, pp. 1521–1537, 2008.

[4] A. Choi, D. Tong, M. K. Jawed, and J. Joo, "Implicit contact model for discrete elastic rods in knot tying," *J. Appl. Mechanics*, vol. 88, no. 5, 2021, Art. no. 051010.

[5] 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.

[6] M. Yu, H. Zhong, and X. Li, "Shape control of deformable linear objects with offline and online learning of local linear deformation models," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 1337–1343.

[7] J. Iqbal, Z. H. Khan, and A. Khalid, "Prospects of robotics in food industry," *Food Sci. Technol.*, vol. 37, pp. 159–165, 2017.

[8] D. Tong, A. Borum, and M. K. Jawed, "Automated stability testing of elastic rods with helical centerlines using a robotic system," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 1126–1133, Apr. 2022.

[9] B. Audoly, N. Clauvelin, and S. Neukirch, "Elastic knots," *Phys. Rev. Lett.*, vol. 99, no. 16, 2007, Art. no. 164301.

[10] M. K. Jawed, P. Dieleman, B. Audoly, and P. M. Reis, "Untangling the mechanics and topology in the frictional response of long overhand elastic knots," *Phys. Rev. Lett.*, vol. 115, no. 11, 2015, Art. no. 118302.

[11] V. P. Patil, J. D. Sandt, M. Kolle, and J. Dunkel, "Topological mechanics of knots and tangles," *Science*, vol. 367, no. 6473, pp. 71–75, 2020.

[12] P. Johanns, P. Grandgeorge, C. Baek, T. G. Sano, J. H. Maddocks, and P. M. Reis, "The shapes of physical trefoil knots," *Extreme Mechanics Lett.*, vol. 43, 2021, Art. no. 101172.

[13] T. G. Sano, P. Johanns, P. Grandgeorge, C. Baek, and P. M. Reis, "Exploring the inner workings of the clove hitch knot," *Extreme Mechanics Lett.*, vol. 55, 2022, Art. no. 101788.

[14] A. Caporali, K. Galassi, R. Zanella, and G. Palli, "FASTDLO: Fast deformable linear objects instance segmentation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9075–9082, Oct. 2022.

[15] 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.

[16] 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.

[17] D. De Gregorio, G. Palli, and L. Di Stefano, "Let's take a walk on superpixels graphs: Deformable linear objects segmentation and model estimation," in *Proc. Asian Conf. Comput. Vis.*, 2018, pp. 662–677.

[18] 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.

[19] 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.

[20] 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/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8573–8581.

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

[22] M. Denninger et al., "BlenderProc2: A Procedural pipeline for photorealistic rendering," *J. Open Source Softw.*, vol. 8, no. 82, p. 4901, 2023.

[23] W. Qiu and A. Yuille, "UnrealCV: Connecting computer vision to unreal engine," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 909–916.

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

[25] 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.

[26] A. Caporali, R. Zanella, D. D. Greogrio, and G. Palli, "Ariadne+: Deep learning–based augmented framework for the instance segmentation of wires," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 8607–8617, Dec. 2022.

[27] A. Caporali, K. Galassi, B. L. Žagar, R. Zanella, G. Palli, and A. C. Knoll, "RT-DLO: Real-time deformable linear objects instance segmentation," *IEEE Trans. Ind. Inform.*, early access, Feb. 16, 2023, doi: 10.1109/TII.2023.3245641.

[28] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231.

[30] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," in *Proc. ACM SIGGRAPH Conf.*, 2008, vol. 63, pp. 1–12.