Towards Manipulation Planning for Multiple Interlinked Deformable Linear Objects

Ankit J. Shah*[1] and Julie A. Shah[1]

Abstract— **Manipulation of deformable linear objects (DLO) has potential applications in aerospace and automotive assembly. The current literature on planning for deformable objects focuses on a single DLO at a time. In this paper, we provide a problem formulation for attaching a set of interlinked DLOs to a support structure through a set of clamping points. We also present a prototype algorithm that generates a solution in terms of primitive manipulation actions. The algorithm guarantees that none of the interlink constraints are violated. Finally, we incorporate gravity in the computation of a DLO shape and propose a property linking geometrically similar cable shapes across the space of cable length and stiffness. This property allows for computation of solutions for unit length and scaling of the solutions to appropriate length, thus potentially making shape computations faster.**

I. Introduction

Often, final assembly lines in the automotive and aerospace industries require working with deformable objects. In this paper, we are concerned with manipulation planning for multiple interlinked deformable linear objects (DLOs). These are deformable objects of a size along one dimension that is much larger than the size along the other two [1],[2]; for example cables or hoses. The specific task considered in this paper is the installation of an interlinked set of cables in an aircraft fuselage. The cables in question are heavy harnesses of wiring that run along the length of the fuselage. The interlinks connecting the cables are fragile and can not bear the weight of the main cables; hence any attempt to manipulate the cables independently without considering these interlinks may result in damage to their electrical integrity.

While many promising approaches have been proposed towards manipulation and motion planning for a single cable with both ends manipulated, to the best of our knowledge, the problem of planning for multiple DLOs has yet to be addressed. In section III, we mathematically define the manipulation planning problem, including task components, transition actions, the structure of a manipulation plan and a validation test for the plan. In section IV, we present a curve optimization formulation to model the shape of the cable in the presence of gravity. We also present a property that defines geometrical similarity between solutions for DLOs of different length and stiffness. In section V, we present an algorithm for planning a sequence of primitive actions to accomplish cable installation, including strategies to resolve any interlink conflicts that may arise. Finally, in section VI, we examine sample solutions developed by the algorithm for installing a set of cables with fewer manipulators than cables.

*Corresponding author: `ajshah@mit.edu`

[1]A. J. Shah and J. A. Shah are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts,

II. Related Work

Prior research in manipulation planning for DLOs has focused on the development of models to enable rapid computation of DLO shape, defining the manipulation tasks, and planning for task execution given a problem definition.

Using minimum-energy-based shape computation is preferred for quasi-static planning, as it directly computes the equilibrium DLO shape. Some works, for instance [3] and [1], frame the problem of shape computation as a non linear optimization problem. Wakamatsu et al. [3] model the configuration parameters along the length of a DLO as a linear combination of basis functions and optimize over the coefficients. Moll and Kavraki [1] use a subdivision-based scheme, with torsion and curvature as the decision variables for optimization. Also, a series of works [2], [4], [5], assess the modeling of a DLO shape as a geometric optimal control problem. Bretl and McCarthy [2] prove that the set of solutions for the curve shape is a finite dimensional smooth manifold, that can be parametrized by a single chart. Mukadam et al. [4] extend this result to multiple grippers along the cable length. Borum et al. [5] prove that the set of all non-intersecting stable configurations is path connected. These results were important in developing some of the motion planning strategies proposed in literature.

Research into planning for deformable linear objects can be classified according to the goals of the manipulation planning task. The most common goal definition is geometric, where the goal shape and position of the DLO are specified. Works including [1], [2] and [4] propose the use of a probabilistic roadmap planner (PRM) with sampling from the domain of either the boundary conditions, or the manifold that describes the set of the solution to the curve minimization problem. Another group of works [6], [7] and [8] propose feedback control schemes to align the DLO with the goal shape. Wada et al. [7] use PID control on a spring-mass lattice-based model. Berenson [8] proposes an approximate Jacobian-based scheme that does not require a physics-based model. More recently, Roussel et al. [9] combined RRT with sampling from the manifold described in [2] with a dynamic simulator, in order to account for contact with the environment.

Another goal specification is where the topological state of the DLO is important, rather than the exact shape – knot tying for instance as explored in works [10], [11], [12],

[13] and [14]. Saha and Isto [13] develop a topological goal definition based on knot theory and propose a PRM-based planner. In [14] they extend the definition and planner to knot tying around objects in the environment.

Finally, the goal state of the DLO may also be defined relative to it's environment. Acker et al. [15], [16] describe the goal state in terms of contact state of the DLO with respect to the environment and construct plans in terms of transitioning contact states. Another work [9] manages to solve routing and positioning problems in highly constrained environments by allowing contact with the obstacles.

In this paper, we provide a formulation for the problem of manipulating a set of interlinked DLOs. We also present a prototype algorithm that generates a plan in the form of primitive manipulation actions that guarantee that all interlink constraints are satisfied, thus connecting the work done in path planning for DLOs to task planning problems at a higher abstraction. Finally, we prove a property that relates geometrically similar cable shapes across different cable lengths and stiffness values, which allows for faster shape computation.

III. Problem Formulation

In this section we define the manipulation planning problem, including the system state, the action types, the interlink constraints and a definition of the structure of a task plan.

A. Task Components

Consider the problem of aligning a network of n_{cables} to a known set of clamping locations. The components that describe this task are the cables, the manipulators, and the interlinks between the cables.

1) **Cables** Each cable includes:

   a) **Cable shape** Each cable's shape is represented by a curve parametrized by the cable length. Each point on the cable maps to a position and the local orientation. Let the i^{th} cable shape be represented by a curve, $\mathbf{C^i}$, where:

   $$\mathbf{C^i}:\ s \rightarrow \mathbb{R}^3 \times S^3 \quad (1)$$

   $$\mathbf{C^i}(s) = \begin{bmatrix} x(s) & y(s) & z(s) & \phi(s) & \theta(s) & \psi(s) \end{bmatrix}^T$$
   $$s \in [0, L^i]; i \in \{1, 2 ... n_{cables}\}$$

   L^i is the length of the i^{th} cable. Additionally the position and orientation are derived as follows: $\mathbf{C^i} = \begin{bmatrix} \mathbf{X^i}(s)^T & \Phi^i(s)^T \end{bmatrix}^T$

   b) **Clamp positions** The cable must be clamped to specified clamping locations, each of which fixes the complete state of the cable at the specified length. The clamp positions for cable i are denoted by an ordered array of K^i where,:

   $$\mathbf{K^i} = \begin{bmatrix} K^i_j \end{bmatrix};\ j \in \{1, ... n^i_{clamps}\};\ K^i_j \in \mathbb{R}^3 \times S^3 \quad (2)$$

   c) **Reference points** Reference points are locations on the cable where it is to be clamped. The number of reference points is equal to the number of clamps, and each reference point is mapped to a corresponding clamp. The reference points for the i^{th} cable are denoted by an ordered array:

   $$\mathbf{R^i} = \begin{bmatrix} r^i_j \end{bmatrix}; j \in \{1, 2, ... n^i_{ref}\}; r^i_j \in \begin{bmatrix} 0, L^i \end{bmatrix} \quad (3)$$

   d) **Gripping Points** A cable can only be gripped by manipulators at certain discrete points along its length called "gripping points". For the i^{th} cable, they are defined as an ordered array $\mathbf{G^i}$:

   $$\mathbf{G^i} = \begin{bmatrix} g^i_j \end{bmatrix}; j \in \{1, 2 ... n^i_{grip}\}; g^i_j \in \begin{bmatrix} 0, L^i \end{bmatrix} \quad (4)$$

2) **Manipulators** Each manipulator configuration is defined by it's position and orientation. If a cable is being grasped by a manipulator, this creates a boundary condition for $\mathbf{C^i}(s)$. The manipulators are part of the set $\mathbf{M}$:

   $$\mathbf{M} = \{M_k\}; k \in \{1, 2, ... n_{manipulator}\} \quad (5)$$
   $$M_k \in \mathbb{R}^3 \times S^3$$

   A manipulator may either be "free" or "occupied." An occupied manipulator is described by the ordered pair (M_k, g^i_j) implying that the k^{th} manipulator is grasping cable i at grip point g^i_j.

3) **Interlink constraints** All interlinks are contained within a set $\mathbf{I}$, with each element is described by a 5-tuple:

   $$\mathbf{I} = \{\left(i_k, j_k, s^i_k, s^j_k, l_k\right)\}; k \in \{1, 2, ... n_{links}\} \quad (6)$$

   $i_k, j_k \in \{1, 2, ... n_{cable}\}; s^i_k \in \begin{bmatrix} 0, L^i \end{bmatrix}; s^j_k \in \begin{bmatrix} 0, L^j \end{bmatrix}; l_k \in \mathbb{R}^+$
   Here, i_k and j_k are the indices of the cables involved in interlink k, s^i_k and s^j_k are the lengths along the respective cables where the interlinks are attached; and l_k is the length of the interlink. An interlink constraint is said to be satisfied when the interlink is not taut.

   $$||\mathbf{X^{i_k}}(s^i_k) - \mathbf{X^{j_k}}(\mathbf{s^j_k})|| < l_k \quad (7)$$

B. System State

The state of the system is defined by the following tuple:

$$\mathbf{S} = \left(\{\mathbf{C^i}\}, \{(K^i_j, s^i_j)\}, \mathbf{M}, \{(M_k, g^i_j, \mathbf{F})\}\right) \quad (8)$$

The elements of the tuple are as follows:

1) $\{\mathbf{C^i}\}$, is the set of curves describing the cable shapes.
2) $\{(K^i_j, s^i_j)\}$, are a set of ordered pairs, describing the clamps used and the points on cable attached to them.
3) $\mathbf{M}$ represents the set of manipulator positions.
4) $(\{M_k, g^i_j)\}$, are a set of ordered pairs describing the occupied manipulators, the cable and the gripping point used to grasp the cable.
5) $\mathbf{F}$, is a binary vector of length n_{links}. The k^{th} element, evaluate the truth value of condition (7)

Figure 1 depicts an example system state. The red lines indicate a taut interlink and the green lines indicate a slack interlink that satisfies the constraint in (7).

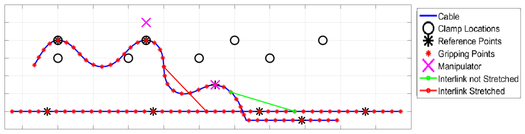

Fig. 1: A visual representation of the System State

C. State Transition Actions

The state transition actions are the set of actions that manipulate the cable. For the scope of this work we assume that a single manipulator can only grasp one cable at a time, and define five action types.

1) **RepositionManipulator**(`ManipID`, `target`)
This action type respositions the selected manipulator to the specified target. `ManipID` $\in \mathbf{M}$ and `target` $\in \mathbb{R}^3 \times S^3$
2) **GraspCable**(`ManipID`, `CableID`, `GripPointID`)
This action results in the selected manipulator moving and grasping the specified cable at the specified gripping point. `ManipID` $\in \mathbf{M}$, `CableID` $= i \in \{1,2,...n_{cable}\}$, `GripPointID` $\in \mathbf{G^i}$
3) **ClampCable**(`ManipID`, `CableID`, `ClampID`)
This action moves the selected manipulator to the clamp position and transfers the boundary value constraint from the manipulator to the selected clamp. (This action requires the manipulator to be grasping the specified cable). `ManipID` $\in \mathbf{M}$, `CableID` $= i \in \{1,2,...n_{cable}\}$ and `ClampID` $\in \mathbf{K^i}$
4) **ReleaseManipulator**(`ManipID`)
This action frees the selected manipulator and removes the boundary condition from the grasped cable. `ManipID` $\in \mathbf{M}$
5) **AlignReferencePoint**(`ManipID`, `CableID`, `RefID`)
This is a compound action. First, a gripping point on the cable closest to the specified reference point is selected, and the action **GraspCable** is used to grasp the cable at this point. Next, the cable is clamped to the corresponding clamp location using the **ClampCable** action.

D. Plan Structure

The manipulation plan is described as the following tuple:

$$\mathbf{P} = \left(\{A_1^1, A_2^1, \ldots\}, \{A_1^2, \ldots\}, \ldots, \{A_1^k, \ldots\}, \ldots \{A_1^n, \ldots\}\right) \quad (9)$$

(9) defines a task plan of length n. Each set included in the tuple is called an *action set*. Each element of an *action set* represents a primitive action. All primitive actions within an *action set* must be executed simultaneously. Further, an *action set* k is executed strictly before $k+1$. In the event that the **AlignReferencePoint** action is carried out in set k, the **GraspCable** action is included in set $k-1$ and the **ClampCable** action is included in k.

In order, for a plan to be valid, none of the primitive actions must result in a taut cable. and no interlink constraints may be violated once all the actions in a give *action set* are completed.

IV. Shape Computation

While planning for an action, it is necessary to predict the cable shape in order to prevent stretching it or violating an interlink constraint. Similar to [3], [1] and [2], we use a minimum-energy-based scheme to compute the shape of the cables. As described in Section III-A, cable shape is represented by a curve defined in Eq (1). The points at which a cable is clamped or grasped by a manipulator define boundary conditions for that cable. The shape of the cable between each of those boundary conditions can be computed independently from the other sections if the length between each point is known. Thus the entire cable shape may be determined by computing a two-point boundary value problem for each section of the cable at a time.

Consider a segment of the cable; let the shape be defined by the curve as follows:

$$\mathbf{C} \ : s \rightarrow \mathbb{R}^3 \times S^3; \ s \in [0,1] \quad (10)$$

$$\mathbf{C}(s) = \begin{bmatrix} x(s) & y(s) & z(s) & \phi(s) & \theta(s) & \psi(s) \end{bmatrix}^T$$

The curve $\mathbf{C}$ must be a solution to the following curve optimization problem.

$$min \ J = \int_0^1 k_1 \frac{u^2}{L^2} + k_2 \frac{\tau^2}{L^2} + LW_g y ds \quad (11)$$

subject to

$$\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{dz}{ds} \\ \frac{d\phi}{ds} \\ \frac{d\theta}{ds} \\ \frac{d\psi}{ds} \end{bmatrix} = \begin{bmatrix} L \ cos(\theta) cos(\psi) \\ L \ cos(\theta) sin(\psi) \\ L \ sin(\theta) \\ \tau + u \ tan(\theta) sin(\phi) \\ u \ cos(\phi) \\ u \ sec(\theta) sin(\phi) \end{bmatrix}$$

with boundary conditions $\mathbf{C}(0) = \mathbf{C_0}$ and $\mathbf{C}(L) = \mathbf{C_f}$

Where k_1 and k_2 are proportional to the bending and torsional stiffness of the cable respectively; W_g is the weighting factor for gravity and u and τ are the curvature and the torsion in the cable respectively.

The problem of shape computation is identical to an optimal control problem and can be solved using calculus of variations by converting the optimization to a two point boundary value problem [17]

The nature of the curve optimization problem allows for its solutions to have the following properties:

1) The solution is invariant with translation.
2) The solution is invariant when the cable is rotated along the gravity vector.
3) The solution for length L and stiffness values k_1 and k_2 are geometrically similar to length L' and stiffness values $\frac{k_1 L'^3}{L^3}$ and $\frac{k_2 L'^3}{L^3}$ given that the boundary conditions are appropriately scaled.

These properties can often be used to improve the speed of shape computation witin planning search. An example of how these properties can be used to compute the solution to a problem with arbitrary boundary conditions and length is depicted in figure 2. This is a 2D problem for computing the shape of a cable with endpoint boundary conditions for $\mathbf{C} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ being $\mathbf{C}(0) = \begin{bmatrix} 0.2 & 0.3 & 0 \end{bmatrix}^T$ and $\mathbf{C}(1) = \begin{bmatrix} 2.7 & 1.3 & 0 \end{bmatrix}^T$. The length of the cable is $L = 5$m and the stiffness value is $k_1 = 0.5$. It is assumed that solution for $L = 1$m and $k_1 = 0.5$ is known (see the dotted blue line in figure 2)

The steps taken to compute the shape are as follows:

1) Provide the known solution as an initial guess to a boundary value problem solver and compute shape for $L = 1$m and $k_1 = 0.004$ with boundary conditions $\mathbf{C}(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and $\mathbf{C}(1) = \begin{bmatrix} 0.5 & 0.2 & 0 \end{bmatrix}^T$ (See the solid blue line in Figure 2)
2) Scale by a factor of 5 (See the dotted red in Figure 2)
3) translate by $\begin{bmatrix} 0.2 & 0.3 \end{bmatrix}^T$ (See the solid red line in Figure 2).

A. *Proof for Length and Stiffness Transformation*

We provide the outline of a proof for the third property, in a 2D case. We use the calculus of variations approach and convert the optimization problem to a boundary value problem. For the original problem with L and k, let $\begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}^T$ be the costates. The boundary value problem that must be solved is given by

$$\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{d\theta}{ds} \\ \frac{dp_1}{ds} \\ \frac{dp_2}{ds} \\ \frac{dp_3}{ds} \end{bmatrix} = \begin{bmatrix} L\ cos\theta \\ L\ sin\theta \\ -\frac{L^2 p_3}{2k_1} \\ 0 \\ -LW_g \\ Lp_1\ sin\theta - Lp_2\ cos\theta \end{bmatrix} \tag{12}$$

subject to the boundary conditions

$$\mathbf{C}(0) = \mathbf{C_0} \text{ and } \mathbf{C}(L) = \mathbf{C_f}$$

The boundary value problem for the upscaled problem of length $L = 1$ with stiffness value $k_1' = \frac{k_1}{L^3}$ is given by:

$$\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{d\theta}{ds} \\ \frac{dp_1}{ds} \\ \frac{dp_2}{ds} \\ \frac{dp_3}{ds} \end{bmatrix} = \begin{bmatrix} L\ cos\theta \\ L\ sin\theta \\ -\frac{L^3 p_3}{2k_1} \\ 0 \\ -W_g \\ p_1\ sin\theta - p_2\ cos\theta \end{bmatrix} \tag{13}$$

subject to the boundary condition

$$\mathbf{C}(0) = \mathbf{C_0} \text{ and } \mathbf{C}(L) = \mathbf{C_f}$$

If we assume that the initial conditions $\begin{bmatrix} x_0 & y_0 & \theta_0 & p_{10} & p_{20} & p_{30} \end{bmatrix}^T$ solves problem (12), then $\begin{bmatrix} x_0 & y_0 & \theta_0 & \frac{p_{10}}{L} & \frac{p_{20}}{L} & \frac{p_{30}}{L} \end{bmatrix}^T$ solves problem 13 with an identical solution for $\begin{bmatrix} x(s) & y(s) & \theta(s) \end{bmatrix}^T$

V. Proposed Planning Technique

The prototype planner proposed here accepts a list of reference points to be aligned, $\{(i, r_j^i)\}$; where $i \in \{1, \ldots, n_{cable}\}$; and $r_j^i \in \mathbf{R^i}$. The output of the planner is a valid task plan in the format defined in (9).

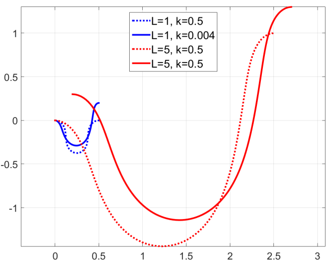

Fig. 2: Example of steps followed for shape computation.

The planner progressively tries to align each reference point in that list (algorithm 1), along with planning actions to resolve any interlink conflicts that might arise. If the planner fails to align the same reference point twice without aligning any new point in between, it exits and returns a failure. The interlink resolution (algorithm 2) can either be achieved with a single-step alignment where another manipulator simultaneously aligns reference points on other cables, or it can be achieved by using the manipulators to position the other cables such that no interlink is taut (algorithm 5). If repositioning is required, the planner determines a new set of reference points (algorithm 4) that allow for freeing of the repositioning manipulators while ensuring that no interlinks are taut. If a plan to align this new list cannot be generated, the resolution is re-attempted using new grip points (algorithm 3).

The proposed algorithm uses an incomplete set of resolution strategies in algorithm 2. However the planning algorithm is otherwise complete within the space of the considered strategies.

A. *Aligning a Reference Point List*

AlignReferencePointList (algorithm 1) takes as input the list of reference points that must be clamped $\{(i, r_j^i)\}$, where $i \in \{1 \ldots n_{cable}\}$, $r_j^i \in \mathbf{R^i}$; the system state $\mathbf{S}$; the task plan $\mathbf{P}$; the list of free manipulators $\{M_{free}\}$, where $M_{free} \in \mathbf{M}$; the list of pairs of occupied manipulators and the cables grasped by them $\{(i, M)\}$, where $i \in \{1 \ldots n_{cable}\}, M \in \mathbf{M}$. The output produced is the final system state and the task plan to transfer the system from input state to output state. In the event that the algorithm fails, it returns a failure flag, along with the system state $\mathbf{S}$ and the task plan $\mathbf{P}$ at the time of the last successful alignment.

Line 4 initializes a list of elements of RefPointList with failed alignment attempts. Line 7 selects an unaligned reference point. In lines 9 and 10, if the attempted alignment of the selected reference point has already failed previously, without a successful alignment in between, the algorithm exits with a failure. Line 12 aligns the selected reference point with it's clamp. From lines 14 to 16, if the alignment does not result in taut interlinks, the FailList is cleared (line 15) and alignment actions are added to $\mathbf{P}$. In the event of violated interlink constraints, line 19 attempts resolution using algorithm 2. If the resolution is successful, Lines 21 and 22 add the alignment and resolution actions to the task plan and clear the FailList, else lines 24 to 26 backtrack $\mathbf{S}$ to the state before the attempted alignment, and add the selected reference point back to the end of the queue. Line 27 declares success and exits if all reference points are aligned.

Algorithm 1 AlignReferencePointList

```
1: function AlignReferencePointList (S, P, FreeManipulators, OccupiedManipulators, RefPointList)
2:     RemainingRefPointList = RefPointList
3:     exitFlag = False; SuccessFlag = False
4:     Initialize FailList
5:     while exitFlag = false do
6:         $M \leftarrow$ Select Manipulator from FreeManipulators
7:         $(i, r^i_j) \leftarrow$ Select from RemainingRefPointList
8:         Delete $(i, r^i_j)$ from RemainingRefPointList
9:         if $(i, r^i_j) \in$ FailList then
10:            exitFlag = True; SuccessFlag = False; Break
11:        else
12:            S.AlignReferencePoint(M,$i$,$r^i_j$)
13:            Determine taut Interlinks
14:            if No Interlinks taut then
15:                Clear FailList
16:                Add alignment actions to P
17:            else
18:                Determine newFreeManipulators and newOccupiedManipulators
19:                (S,P,ResolveFlag) = ResolveInterlinkConflict(S,P, newOccupiedManipulators,newFreeManipulators,$(i, M)$)
20:                if ResolveFlag == true then
21:                    Clear FailList
22:                    Add alignment and resolution actions to P
23:                else
24:                    Add $(i, r^i_j)$ to FailList
25:                    Backtrack S to before alignment
26:                    Add $(i, j)$ to end of RemainingRefPointList
27:        if isEmpty(RemainingRefPointList) then
28:            exitFlag = True; SuccessFlag = True
29:            break
30:    return S, P, SuccessFlag
```

B. Interlink Conflict Resolution

Algorithm 1 calls **ResolveInterlinkConflict** (algorithm 2) whenever an alignment results in taut interlinks. **ResolveInterlinkConflict** takes as input the system state **S** and the task plan **P** that involves the attempted reference point alignment. Additionally, the input includes the free manipulators, $\{M_{free}\}$, where $M_{free} \in \mathbf{M}$; the list of tuples of occupied manipulators, $\{(M, g^i_j)\}$, where $i \in \{1 \dots n_{cable}\}$, $M \in \mathbf{M}$, $g^i_j \in \mathbf{G^i}$; and ClampedCable $(i_{clamped}, M_{clamped})$, a tuple including the cable being clamped and the manipulator clamping it $i_{clamped} \in \{1 \dots n_{cable}\}$, $M_{clamped} \in \mathbf{M}$. The resolution is considered successful if a system state **S** and a valid task plan **P** are found such that, in **S**, no interlinks are taut and the manipulators in FreeManipulators are free. If not, the algorithm fails and returns the **S** and **P** provided at input.

Line 2 creates CorrespondingCable, a set of tuples $\{(i, \mathbf{k}, \{r\}, \{g\})_a\}$, where $a \in \{1 \dots n_{correspCables}\}$, $i \in \{1 \dots n_{cable}\}, \mathbf{k} \subseteq \mathbf{I}, \{r\} \subseteq \mathbf{R^i}; \{g\} \subseteq \mathbf{G^i}$. This is a set of all cables attached to at least one taut interlink. i represents the index of the cable, **k** is the set of taut interlinks attaching to cable i. $\{r\}$ is the set of reference points that lie between the first reference points to the right and left of all taut interlinks on the cable. Figure 3 (a) depicts an instance of a taut interlink and reference points in $\{r\}$. $\{g\}$ is a set of grip points that will be considered for grasping the cable in order to reposition it. As shown in Figure 3 (b) all gripping points between the first slack interlinks to the right and left of the taut interlinks are included. In lines 3 to 5, the algorithm declares failure and exits if the number of free manipulators is less than the number of ungrasped corresponding cables.

Algorithm 2 ResolveInterlinkConflict

```
1: function ResolveInterlinkConflict (S, P, FreeManipulator, OccupiedManipulators, $(i_{clamped}, M_{clamped})$)
2:     Initialise CorrespondingCables
3:     if size(CorrepondingCables) - size(OccupiedManipulators) > size(FreeManipulators) then
4:         SuccessFlag = False;
5:         return SystemState, TaskPlan, SuccessFlag
6:     for all CorrespondingCables do
7:         if $(i, \mathbf{k}, \{r\}, \{g\}) \in$ CorrespondingCables $\exists$ $M$ such that $(M, g^i_j) \in$ OccupiedManipulators then
8:             CorrespondingCablesManip.Add($i, \mathbf{k}, \{r\}, \{g\}, M$)
9:         else
10:            $M \leftarrow$ First element of FreeManipulators
11:            Delete $M$ from FreeManipulators
12:            CorrespondingCablesManip.Add($i, \mathbf{k}, \{r\}, \{g\}, M$)
13:    if $\exists$ $i$ such that $(i, \mathbf{k}, \{r\}, \{g\}, M)$ CorrespondingCablesManip and $(M, g^i_j) \notin$ OccupiedManipulators then
14:        Attempt Single Step Resolution and create SingleStepList and RepositionList
15:        Add Single step resolution actions to P
16:    if No interlink is strethced then
17:        SuccessFlag = True
18:        return S, P, SuccessFlag
19:    (S, P, RepositionFlag) = AttemptRepositionResolution(S, P, RepositionList, CorrespondingCablesManip)
20:    if ResolveFlag == True then
21:        SuccessFlag = True;
22:        Add reposition actions to P
23:        return S, P, SuccessFlag
24:    else
25:        SuccessFlag = False;
26:        Backtrack S
27:        return S, P, SuccessFlag
```

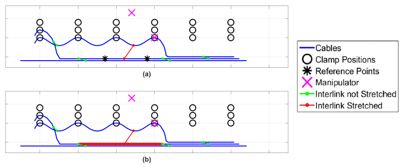

Fig. 3: An instance of a violated interlink and the reference points and gripping points considered

Lines 6 to 12 ensure that a manipulator is assigned to each of the corresponding cables. Line 14 attempts a single step resolution and determines feasible single step actions. Single-step resolution is feasible if the corresponding cable can be clamped at one of the reference points in $\{r\}$ while resolving the taut interlinks and not stretching any other interlinks on the cable. As we are only concerned with interlinks on the selected corresponding cable and the clamped cable, we search through the reference points on the corresponding cables independently. We do not perform a full combinatorial search for all reference points and assigned manipulators Cables for which a single step resolution is not feasible are added to the RepositionList, a set of cables that will be repositioned in order to resolve any remaining interlink constraints. Lines 16 to 18 exit after declaring success if the interlinks are resolved with only single step resolution. Line 10 uses algorithm 3 to plan a repositioning based resolution strategy. Lines 20 to 27 exits the algorithm with a flag indicating repositioning success or failure.

Next, we consider the function **AttemptRepositionResolution** (algorithm 3). This function is called in algorithm 2 in the event that single step resolution does not succeed. It accepts as input the system state **S**, the task plan **P** and the CorrespondingCablesManip set of tuples from algorithm

Algorithm 3 AttemptRepositionResolution

```
1: function AttemptRepositionResolution(S, P, RepositionList, CorrespondingCablesManip)
2:     if ∃ $M$ such that $(i,\mathbf{k},\{r\},\{g\},M) \in$ CorrespondingCablesManip and $M \in$ FreeManipulators then
3:         NewRefPointList = DetermineReferencePointsToAlign (S, CorrespondingCablesManip, RepositionList)
4:     exitFlag = False; SuccessFlag = False; GripListEmptyCount = 0;
5:     while exitFlag == False do
6:         if GripListEmptyCount ≥ n(CorrespondingCablesManip) - n(OccupiedManipulators) then
7:             exitFlag = True; SuccessFlag = False; break
8:         GripListEmptyCount = 0
9:         for $\forall(i,\mathbf{k},\{r\},\{g\},M) \in$ CorrespondingCablesManip such that $i \in$ RepositionList & $m \in$ FreeManipulators do
10:            if $\neg IsEmpty(\{g\})$ then
11:                $g^i_j \leftarrow$ Sample grip point from $\{g\}$
12:                S.GraspCableParallel($m,i,g^i_j$)
13:                Delete $g^i_j$ from $\{g\}$
14:            else
15:                Increment GripListEmptyCount
16:        (S, P, RepositionFlag) = ComputeGeometricResolution(S, P, CorrespondingCablesManip, RepositionList)
17:        if RepositionFlag == False then
18:            continue
19:        else
20:            Determine NewFreeManipulators, NewOccupiedManipulators
21:            (S, P, AlignFlag) = AlignReferencePointList(S, P, NewFreeManipulators, NewOccupiedManipulators, NewRefPointsList)
22:            if AlignFlag == True then
23:                exitFlag = True; SuccessFlag = True
24:            else
25:                continue
26:    return S, P, SuccessFlag
```

2. It also requires RepositionList $\{i\}$, a list of cables that need to be repositioned as determined in algorithm 2, where $i \in \{1 \dots n_{cable}\}$. If successful, this algorithm computes the cable positions that ensure that the interlink constraints are resolved. It also determines the task plan for aligning a subset of reference points that ensure the interlink constraints are satisfied even after the manipulators that were in FreeManipulators at input to algorithm 2 are released.

From lines 2 to 3 the additional set of reference points that must be aligned are determined using algorithm 4. For each free manipulator, there exists a list of grip points that must be tried to compute a resolution for the interlink constraints. Lines 6 to 7 declare failure and exit if no new grip points are available to check for resolution. Between lines 10 and 15, a grip point is selected from the set $\{g\}$, for each corresponding cable with a free manipulator assigned to it. Next, line 16 computes a geometric resolution using algorithm 5. In the event that algorithm 5 fails, the process repeats with a selection of new grip points. If successful, lines 20 to 21 attempt to align the list of reference points determined in line 3 using algorithm 1. Finally lines 22 to 25 declare success and exit. (If the alignment of the reference point list fails, the algorithm continues.)

Next, we look at the function **DetermineRefPointsToAlign**(algorithm 4). This algorithm takes as input the system state **S** after a reference point alignment. It also accepts the set of tuples CorrespondingCablesManip and the RepositionCables list from algorithm 2. Lastly, as the function has been designed as a recursive algorithm, it also accepts an initial alignment list of reference points: $\{(i,r^i_j)_m\}$, where $i \in \{1,\dots,n_{cable}\}; r^i_j \in \mathbf{R^i}$. The reference

Algorithm 4 DetermineRefPointsToAlign

```
1: function DetermineRefPointsToAlign( S, CorrespondingCables, InitialList, Manip)
2:     FinalList = InitialList
3:     for all $(i,\mathbf{k},\{r\},\{g\}) \in$ CorrespondingCables & $i \in$ RepositionList do
4:         Sort elements of k by priority function
5:         m=0
6:         while exitFlag == False do
7:             $r^i_j \leftarrow m_{th}$ element of k
8:             SystemState.AlignRefPoint(Manip, $i$, $r^i_j$)
9:             Increment m
10:            Add $(i,r^i_j)$ to FinalList
11:            if All interlinks in j are resolved then
12:                exitFlag = True;
13:        if Interlinks$notin$ j are violated then
14:            Initialise set NewCorrepondingCables
15:            FinalList = DetermineRefPointsToAlign(SystemState, NewCorrespondingCables, FinalList, Manip)
16:    return FinalList
```

point alignments in this function exist only in planner memory; they are not actually executed. This algorithm determines the list of reference points that must be aligned to ensure that all interlink constraints are satisfied even after the manipulators used to reposition the cables are freed.

Line 2 ensures that the InitialList is included in the final list. A list of reference points for each corresponding cable must be determined. The relevant reference points are aligned in the ascending order of the priority function:

$$f(r^i_j) = \sum_{(i,a,s^i,s^a,l)\in\mathbf{j}} |s^i - r^i_j|; r^i_j \in \mathbf{k} \tag{14}$$

The priority function is the summation of the absolute distance of the selected reference point from all the taut interlinks along the length of the cable. This ensures that the reference point cumulatively closest to the attachment points on all interlinks receives the highest priority. Lines 6 to 12 align the reference points according to the priority function, until all of the originally violated interlink constraints are resolved. The function exits if no new interlink constraints are violated; otherwise, a new tuple set of CorrespondingCables is created for the cable under consideration in line 14. The function is recursively called in line 15 while providing the alignment list determined thus far as an input.

C. Computing a Resolution to the Interlink Conflict

In the event that single-step resolution in algorithm 2 fails, the other cables must be repositioned to ensure that no interlink is taut. This is a planning problem in continuous space of manipulator positions. In order to address this problem, we use a technique similar to the that described by Berenson in [8].

Consider a cable j being repositioned using manipulator l with grip point m at length g^j_m. Let $M_l \in \mathbf{M}$ be the configuration of the manipulator and $\mathcal{P}$ be the vector denoting the concatenated positions of P attachment points of the taut interlinks on the manipulated cable. Hence $\mathcal{P} \in \mathbb{R}^{3P}$. In order to resolve the interlink conflicts, these points must be lifted to the points on the cables at the other end of the taut interlinks. Let the concatenated positions of these target points be specified in $\mathcal{T} \in \mathbb{R}^{3P}$ and, let the function $F(M_l): \mathbb{R}^3 \times S^3 \to \mathbb{R}^{3P}$ map the manipulator configuration to the positions of the points in $\mathcal{P}$:

$$\begin{aligned}\mathcal{P} &= F(M_l)\\ \dot{\mathcal{P}} &= \frac{\partial F}{\partial M_l}\dot{M}_l = J\dot{M}_l\end{aligned} \tag{15}$$

Berenson [8] uses an approximation of the Jacobian to speed up computation. For each point, the Jacobian is approximated as the rigid body Jacobian weighted by a "rigidity factor":

$$\mathbf{J_i} = w_i \mathbf{J_{i_{rigid}}}; i \in \{1,2,\dots P\} \tag{16}$$

$$\mathbf{J} = \begin{bmatrix}\mathbf{J_1}^T & \mathbf{J_2}^T & \dots & \mathbf{J_i}^T & \dots & \mathbf{J_P}^T\end{bmatrix}^T \tag{17}$$

Where w_i is the "rigidity factor" for the i^{th} point and is computed as follows:

$$w_i = e^{-k|g_m^j - s_{p_i}^j|} \tag{18}$$

Here, k is a tunable parameter that controls the rigidity of the cable and $s_{p_i}^j \in [0, L^j]$ is the lengthwise position of the point considered along the cable. The rigidity factor is unity at the gripping point and gradually decays further along the cable. This represents the diminishing effect of the movement further away from the gripping point.

J_{rigid} is the rigid body Jacobian and is given by:

$$\mathbf{J_{i_{rigid}}} = \begin{bmatrix}\mathbf{I_3} & \mathbf{J_{i_{rot}}}\end{bmatrix} \tag{19}$$

$$J_{i_{rot}} = [r]_\times^T \mathbf{T_{IM}} \mathbf{A} \tag{20}$$

Where $\mathbf{T_{IM}}$ is the transformation from the manipulator frame to the ground frame, $r = \mathbf{X}(s_{p_i}^j) - \mathbf{X}(M_l)$ is the vector from manipulator position to the point. $[r]_\times$ is the skew-symmetric cross product matrix and $\mathbf{A}$ is defined as

$$\mathbf{A} = \begin{bmatrix}1 & 0 & -sin\theta\\ 0 & cos\phi & sin\phi cos\theta\\ 0 & -sin\phi & cos\phi cos\theta\end{bmatrix} \tag{21}$$

The control input applied to the manipulator are computed as follows

$$\dot{M}_l = \mathbf{J}^\dagger(\mathcal{T} - \mathcal{P}) \tag{22}$$

where $J^\dagger$ denotes the Moore-Penrose inverse of J.

The complete algorithm for computing the resolution is described in algorithm 5. **ComputeGeometricResolution** takes as input the system state **S** and the task plan **P**. In **S**, the corresponding cables have already been grasped at grip points sampled in algorithm 3. Additionally, CorrespondingCablesManip and RepositionCables from algorithm 2 must also be provided as input. If successful, the output includes **S** with the cables repositioned such that no interlinks are taut, and the necessary actions are added to **P**. Else, **S** and **P** at input are returned along with, a binary success flag is also returned.

Line 3 intializes, $\mathcal{P}$ and $\mathcal{T}$ using the attachment points of taut interlinks. If additional interlinks become taut over the course of computing the resolution positions, line 7 adds their attachment points to $\mathcal{P}$ and $\mathcal{T}$. Lines 9 and 10 propagate the position of the manipulator according to equation 22 and reposition the manipulator, respectively. Lines 11 to 12 exit the algorithm with a failure if the maximum number of iterations is exceeded without resolution. Lines 13 to 15 exit the algorithm declaring success if the interlink constraints are satisfied.

Algorithm 5 ComputeGeometricResolutions

```
1:  function COMPUTEGEOMETRICRESOLUTION (S, P, CorrespondingCablesManip, RepositionCables)
2:      for all $(i,\mathbf{k},\{r\},\{g\},M) \in$ CorrespondingCablesManip such that $i \in$ RepositionList do
3:          Initialize $\mathcal{P}$ and $\mathcal{T}$ vectors
4:      while exitFlag == false or iterCount $\le$ MaxIter do
5:          for all $(i,\mathbf{k},\{r\},\{g\},M) \in$ CorrespondingCablesManip such that $i \in$ RepositionList do
6:              if Interlinks on $i$ are taut then
7:                  Add attachment points of new taut interlinks to $\mathcal{P}$ and $\mathcal{T}$
8:                  J ← Compute Jacobian
9:                  $\delta M_k$ ← Propagate control
10:                 S.RepositionManipulator($k$, $M_k + \delta M_k$)
11:             if iterCount $\ge$ MaxIter then
12:                 SuccessFlag = False
13:             if no interlinks on $i$ strethced then
14:                 exitFlag = True; SuccessFlag = True
15:                 Add reposition actions to P
16:     return S, P, SuccessFlag
```

D. Analysis

The worst case time complexity of the planning algorithm is $\mathcal{O}(N_{ref}^{2N_{manip}} N_{grip} + N_{ref}^{2N_{manip}+1})$. Where $N_{ref}, N_{grip}, N_{manip}$ are the total numbers of reference points, grip points and manipulators respectively. Single step resolution involves at most $\mathcal{O}(N_{ref} * N_{manip})$ steps. Resolution by respositioning involves $\mathcal{O}(N_{grip} * N_{manip})$ attempts. Determining the subset of reference points to align has the worst case complexity of $\mathcal{O}(N_{ref})$. As the algorithm recurses, the number of available manipulators N_{manip} is reduced by at least one. Thus the recursion is at most N_{manip} deep.

VI. Example Solutions

In this section, we describe sample solutions for a manipulation planning problem involving three cables and two manipulators. The interlinks are placed on the cables in such a fashion that it is possible to align them using only two manipulators. The initial state of the problem is depicted in Figure 4. We present two instances of resolution strategies developed by the planner. The entire plan is depicted in accompanying video. (https://youtu.be/MkU1lcfIwDQ)

We begin by attempting to align cable 1 at reference point 1. The interlink constraint violated by the alignment is depicted in Figure 5 (a). In this case, it is possible to perform a single step alignment by simultaneously aligning cable 2 at reference point 1 as depicted in Figure 5 (b).

Another instance is the alignment of cable 1 at reference point 6. The interlink constraint violated as a result of that alignment is depicted in Figure 6. In this case, single step alignment is not possible, as shown in Figure 7; therefore cable 2 must be repositioned in order to resolve the interlink constraint. Additionally, the following set of cables and reference points, respectively, must be aligned to ensure that the no interlink constraints are violated $\{(2,2),(3,4),(2,5)\}$. The state after resolving the interlink constraint by repositioning cable 2 is shown in Figure 8. The first reference point of the aforementioned list that must be aligned is $(2,2)$,

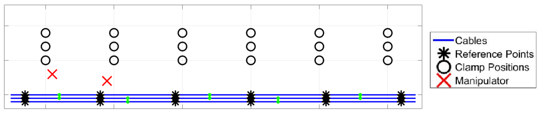

Fig. 4: Initial State of the problem

as aligning any other beforehand would result in additional violated constraints without having the free manipulators necessary to resolve them. $(2,2)$ does not violate any new constraints as shown in Figure 9 (a). Once $(2,2)$ is aligned, $(3,4)$ can be aligned next; however cable 2 needs to be repositioned again to ensure that the interlink between cables 3 and 2 is not taut during this alignment. Finally, once $(3,4)$ is aligned, $(2,5)$ can be aligned. No action is required in this case as this alignment will not violate any constraints.

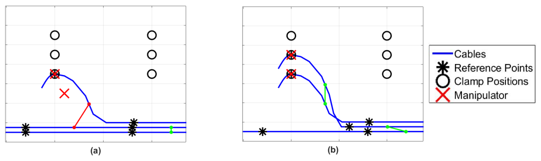

Fig. 5: Example single step resolution

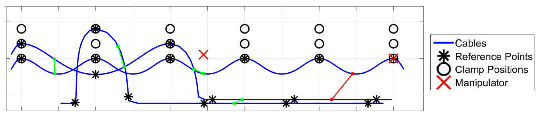

Fig. 6: An interlink violated during reference point alignment

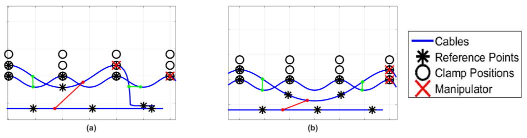

Fig. 7: Infeasible Single step resolutions

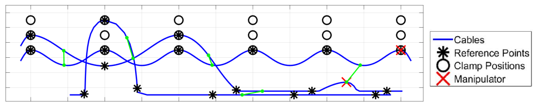

Fig. 8: Interlink violation resolved through a cable repositioning

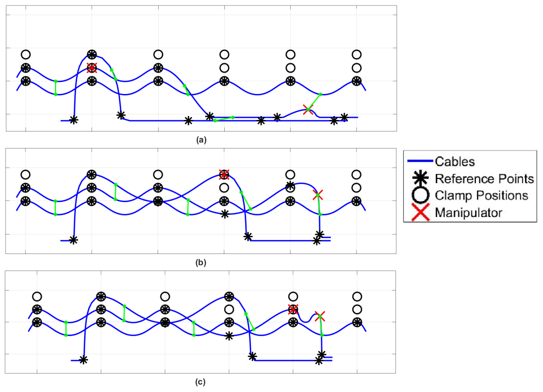

Fig. 9: Alignment of additional reference points to ensure resolution

VII. Summary and Future Work

In summary, we present a model for posing a manipulation planning problem with multiple interlinked DLOs, including mathematical descriptions for the system state, transition actions and a plan validator. We also present a planning algorithm that generates the plan in terms of primitive actions to align a given set of reference points without violating any interlink constraints. Finally, we present, with proof, a scaling transformation that maps similar DLO shapes across varying length and stiffness values.

The proposed planner is complete with respect to the considered set of resolution strategies, but our proposed set is not complete. We assume that each cable is manipulated by, at most, one manipulator at a time. Consider a scenario with a number of taut interlinks attached to a given manipulated cable, such that there exists a clamped point between one of the taut interlinks and the location at which the cable is grasped. While the position of the attachment point cannot be influenced by the occupied manipulator, a second free manipulator could resolve the taut interlink. In future work we will investigate the development of a complete planner that necessarily reasons over such distinct segments of the cables, to provide a complete set of resolution strategies.

References

[1] M. Moll and L. E. Kavraki, “Path planning for deformable linear objects,” *TRO*, vol. 22, no. 4, pp. 625–636, 2006.
[2] T. Bretl and Z. McCarthy, “Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations,” *IJRR*, vol. 33, no. 1, pp. 48–68, 2014.
[3] H. Wakamatsu and S. Hirai, “Static modeling of linear object deformation based on differential geometry,” *The International Journal of Robotics Research*, vol. 23, no. 3, pp. 293–311, 2004.
[4] M. Mukadam, A. Borum, and T. Bretl, “Quasi-static manipulation of a planar elastic rod using multiple robotic grippers,” in *Proc. of IROS*, pp. 55–60, IEEE, 2014.
[5] A. Borum and T. Bretl, “The free configuration space of a kirchhoff elastic rod is path-connected,” in *Proc. of ICRA*, pp. 2958–2964, IEEE, 2015.
[6] S. Hirai and T. Wada, “Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model,” *Robotica*, vol. 18, no. 01, pp. 3–11, 2000.
[7] T. Wada, S. Hirai, S. Kawamura, and N. Kamiji, “Robust manipulation of deformable objects by a simple pid feedback,” in *Proc. of ICRA*, vol. 1, pp. 85–90, IEEE, 2001.
[8] D. Berenson, “Manipulation of deformable objects without modeling and simulating deformation,” in *Proc. of IROS*, pp. 4525–4532, IEEE, 2013.
[9] O. Roussel, A. Borum, M. Taix, and T. Bretl, “Manipulation planning with contacts for an extensible elastic rod by sampling on the submanifold of static equilibrium configurations,” in *Proc. of ICRA*, pp. 3116–3121, May 2015.
[10] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, “Knot planning from observation,” in *Proc. of ICRA*, vol. 3, pp. 3887–3892, IEEE, 2003.
[11] H. Wakamatsu, A. Tsumaya, E. Arai, and S. Hirai, “Planning of one-handed knotting/raveling manipulation of linear objects,” in *Proc. of ICRA*, vol. 2, pp. 1719–1725, IEEE, 2004.
[12] A. M. Ladd and L. E. Kavraki, “Motion planning for knot untangling,” in *Algorithmic Foundations of Robotics V*, pp. 7–23, Springer, 2004.
[13] M. Saha and P. Isto, “Motion planning for robotic manipulation of deformable linear objects,” in *Proc. of ICRA*, pp. 2478–2484, IEEE, 2006.
[14] M. Saha and P. Isto, “Manipulation planning for deformable linear objects,” *TRO*, vol. 23, no. 6, pp. 1141–1150, 2007.
[15] J. Acker and D. Henrich, “Manipulation of deformable linear objects: From geometric model towards program generation,” in *Proc. of ICRA*, pp. 1541–1547, IEEE, 2005.
[16] J. Acker, B. Kahl, and D. Henrich, “Environment guided handling of deformable linear objects: From task demonstration to task execution,” *VDI BERICHTE*, vol. 1956, p. 231, 2006.
[17] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2012.