Planning for Manipulation of Interlinked Deformable Linear Objects With Applications to Aircraft Assembly

by

Ankit Jayesh Shah

B.Tech., Indian Institute of Technology Bombay (2013)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2016

Signature redacted

Author .....
Department of Aeronautics and Astronautics
May 19, 2016

Signature redacted

Certified by .....
Julie A. Shah
Associate Professor
Thesis Supervisor

Signature redacted

Accepted by .....
Paulo Lozano
Chairman, Department Committee on Graduate Theses

Planning for Manipulation of Interlinked Deformable Linear Objects With Applications to Aircraft Assembly

by

Ankit Jayesh Shah

Submitted to the Department of Aeronautics and Astronautics
on May 19, 2016, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Manipulation of deformable linear objects (DLO) has potential applications in the fields of aerospace and automotive assembly. In this paper, we introduce a problem formulation for attaching a set of interlinked DLOs to a support structure using a set of clamping points. The formulation describes the manipulation planning problem in terms of known clamping locations; pre-determined ideal clamping locations on the cables, called "reference points", and a set of finite gripping points on the DLOs. We also present a prototype algorithm that generates a solution in terms of primitive manipulation actions. The algorithm guarantees that no interlink constraints are violated at any stage of manipulation. 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 the computation of solutions for unit length and scaling of these solutions to appropriate length, potentially resulting in faster shape computation.

Thesis Supervisor: Julie A. Shah
Title: Associate Professor

Acknowledgments

I would like to thank Prof. Julie Shah for being an amazing advisor and a mentor to me for the past couple of years. Her mentorship and encouragement have been crucial in my growth as a researcher since I started out in my graduate school journey. Through various opportunities that she has provided me with, I have understood the importance of salient and lucid communication of research material.

I would also like to thank my collaborators Ivan Garcia, Diego Arnaut, Nelson Macedo and Matheus de Melo for providing timely constructive feedback and crucial industrial experience to the project. I would also like to thank Nathaniel Rodman and David Perez, who have worked with me during the course of this project. I am very grateful to Beth Marois, Marie Stuppard, Liz Zotos and Julia Finn from the Department of Aeronautics and Astronautics for ironing out all administrative difficulties that I have ever had.

My labmates at the Interactive Robotics Group (IRG) have been an amazing help with ever present technical help, useful pointers and domain expertise in their very diverse research areas. It is my privilege to know them as professional colleagues and personal friends. My classmates from the different controls, estimation and mathematics classes that I took over the years have been a big part of the process of learning, critique and improvement for the doctoral qualifying exams.

The last couple of years would surely have been more monotonous if not for my amazing friends,"The Junta" (Saurabh, Reetik, Chaitanya, Sami, Shweta, Kairav, Aditya, Rahul and Swapn) and many others in Tang, Sid-Pac, Ashdown, Edgerton and The Warehouse and the Boston-Cambridge-Somerville area. I would specially like to thank my childhood friend Maya for her constant support through the last few months of constant writing.

Last but not the least, I thank my parents for believing in me for as long as I have had memories. Despite being in the other hemisphere, I know that I always have their and my family's support.

Contents

List of Figures

Chapter 1

Introduction

Final assembly in aerospace and automotive manufacturing have traditionally been manual processes with very little automation. Final assembly is largely considered to be an "artisanal" task requiring fine dexterity in manipulating work pieces and tools. It is an unstructured domain with each unit being subtly different. The exact sequence of tasks for final assembly is usually ill-defined with each worker having their own preference for task sequencing. Finally, many tasks might require ad hoc teaming of workers to complete successfully. Currently, automation has successfully been deployed in tasks that are highly repetitive in nature. The robots are programmed to perform a pre-planned sequence of actions on each work piece with all variations being explicitly programmed. Such an approach to robot control cannot cope with the unstructured domain of final assembly. Algorithms that allow robots to function in domains subject to various perturbations from nominal are an important challenge in robotics research.

One of the challenges faced in automating final assembly tasks is working with deformable objects like cables, foam pads and rubber hoses. These objects change their shape when an external manipulating force is applied to them. A robot manipulating a deformable object must plan to accomplish the specified task while accounting for the deformations it's actions might induce on the object.

The specific final assembly task that we wish to tackle in this thesis is installation of electrical cables in an aircraft fuselage. Multiple electrical cable harnesses run

through the length of the fuselage on both sides. These harnesses are interlinked at certain points along their length by smaller electrical wires. The interlinks are much smaller than the main cable harnesses and are incapable of bearing the weight of the larger harnesses, thus care must be taken to never stretch the interlink during the process of installation. Currently, this task is typically performed by a large team of workers in three steps. First the cables are laid on the fuselage floor close to the clamping locations and then are aligned with their clamping locations using temporary fasteners. Next, the minor branches that exit the main harnesses are secured. Finally, the main harnesses are secured using the permanent clamps. The initial alignment is prone to errors which are only discovered down the line and are time consuming to correct. With good sensing capabilities, a mobile robot might be used to accomplish the initial alignment of the cables with a greater accuracy than human workers. For that the robotic system must be capable of planning the manipulation of the interlinked cable harnesses without stretching any of the interlinks. In robotics parlance, Electrical cables are classified as deformable linear objects (DLOs). Motion planning for DLOs has been an active area of robotics research.

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. The state of the art methods for motion planning with DLOs operate in the continuous state space of DLO shape and manipulator position and work for a single DLO. Manipulation planning for multiple interlinked is inherently a mixed discrete and continuous problem. The cable shape and manipulator positions lie in a continuous state space, whereas the actions of clamping the cable to pre-defined clamp location, grasping or releasing a cable may be modeled as binary predicates forming a discrete state space. This can be tackled in a hierarchical manner by defining a higher level task planner and a lower level motion planner. In this thesis, we propose the problem formulation for the higher level task planner and a prototype planner to generate satisficing solutions to the task planning problem. A novel feature of the planning algorithm is that, in addition to solving the task planning problem, it also provides

geometric end points for the motion planning problem to connect the discrete planner states. This thesis is largely based on our previously published work [47].

In chapter 2, we present the curve optimization framework used to compute the shape of the cables in response to clamping and manipulator actions. We present the conditions under which the cable shape solutions are equivalent or geometrically similar across different cable lengths and stiffness values. In chapter 3, we provide the mathematical formulation for a manipulation planning problem with multiple interlinked DLOs. This includes the task components, transition actions, the structure of a manipulation plan and a test for evaluating the feasibility of a given manipulation plan. In chapter 4, we present our algorithm for planning a sequence of primitive actions to accomplish cable installation, including strategies to resolve any interlink conflicts that may arise. In chapter 5, we examine sample solutions developed by the algorithm for installing a set of cables on a synthetic example to demonstrate the working of the algorithm in detail. We demonstrate the efficacy of the algorithm on a real-world cable harness installation problem. Finally, we derive the worst case time complexity of the proposed algorithm and describe the challenges involved in designing a complete algorithm.

1.1 Related Work

The current research in manipulation planning for deformable linear objects (DLOs) can be classified into, computational models for predicting the shape of a DLO; paradigms of problem formulation focusing on state and goal definition; and planning methods to solve the planning problems. The models for shape prediction allow the planner to plan proactively by providing a prediction of the DLO shape in response to an applied manipulation action. The different problem formulations differ in the state description based on the goal to be achieved, be it a geometrically defined goal where the entire goal cofiguration of the DLO is defined or a knot tying problem where many different geometric shapes might be equally valid goal states. This section provides an overview of the prior research in each of these areas.

1.1.1 DLO Shape Models

Methods for modeling the shape of a DLO, studied in prior works can be classified into non-physical and physical methods. Non-physical methods aim to generate curves that mimic DLOs only visually. These techniques are not grounded in physical phenomena that govern the behavior of DLOs. By contrast, physical methods, as the name suggests are grounded in the mechanics and dynamics that govern the behavior of objects like cable, hose pipes or surgical wires. They provide higher fidelity and are suited for applications where physical interactions accompany simulations.

Non physical methods rely on continuous and differentiable curves like splines and active contours to generate curves that visually mimic DLOs. Works by Piegl [42], Feng et al. [13] and Chang et al. [9] used splines, and their planar and 3D extensions, to allow a user to set curve positions and derivatives at each of it's control points. The shape of the curve is manipulated by moving the control points as desired. Such methods have been used in applications like computer graphics [10] and garment design [17]. Kass et al. [24] used active contours to discretize a curve into many segments which can reconfigured in order to minimize a cost function which penalizes non-differentiability. These methods are best suited for graphics and virtual reality applications where true physical fidelity is not critical.

Amongst physical methods, is a class of methods that simulates the dynamics of a DLO restricted by boundary conditions and subjected to manipulation forces. These methods discretize the DLO into a series of connected dynamic elements and compute the derivative of the configuration of the DLO. With the knowledge of external forces on the DLO, the dynamic simulation is propagated forward in time to predict the DLO configuration. Discretizations proposed in prior literature vary from interlinked spring-mass-damper models (Brown et al. [8], Li et al. [29], Loock et al. [30]), to more sophisticated semi-continuum models (Wakamatsu et al. [53],[54], Pai [37]), to a full scale finite element method (FEM) simulations (Cohen et al. [11], Müller et al. [36] and Teropoulos [49]). However, in many assembly operations, the DLOs are moved at a speed slow enough that the dynamic movements of the DLOs can

be ignored. In these applications it is important to know the shape of the cable in instances where it is held at a constant position for a period of time. In these instances, the dynamic model needs to be driven to equilibrium. Here, tuning of the dynamic model parameters is crucial to achieve quick convergence. In addition, the time derivative computations at each step are seen as unnecessary, when only the equilibrium state is required.

In instances where the DLO is always in equilibrium or moved slow enough to assume a quasi static process, direct computation of the steady state is beneficial. Minimum energy formulations combined with curve optimization schemes aim to do just that. In equilibrium, the DLO assumes a shape that minimizes the total potential energy of the system. This would typically include the gravitational and the elastic potential energy. Some works, for instance Kallay [23], Wakamatsu et al. [52], Moll and Kavraki [33] and Bergou et al. [4] framed the problem of shape computation as a non linear optimization problem. Wakamatsu et al. [52] modeled the configuration parameters along the length of a DLO as a linear combination of basis functions and optimized over the coefficients. Moll and Kavraki [33] used a subdivision-based scheme, with torsion and curvature as the decision variables for optimization. Javdani et al. [22] used an optimisation-based modeling scheme in a perception system to detect and localise DLOs in space. Another series of works assessed the modeling of a DLO shape as a geometric optimal control problem. Bretl and McCarthy [7] proved 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. [35] extended this result to multiple grippers along the cable length. Borum et al. [6] proved that the set of all non-intersecting stable configurations is path connected. These results were important in the development of some of the motion planning strategies proposed in prior literature.

1.1.2 Goal State Formulations

The goals for manipulating a DLO can be manifold, from twisting a section of rope around a structure, to clamping electrical cables, to tying surgical knots. The planning

formulation and algorithm best suited for the application depends on the description of goal conditions that need to be satisfied. A large body of literature is dedicated to defining planner states and goal conditions for different applications.

The most common definition involves a complete geometric description of the goal state. In such a definition, the position of every point on the DLO in the goal state must be defined. Majority of the prior literature in path planning for DLOs, for instance the works by Moll and Kavraki [33], Bretl and McCarthy [7], Wada et al. [51], Berenson [3], considered planning in the geometric paradigm. While the geometrical goal state is simple from the planning point of view, it is difficult to define a complete configuration for many goal states. In processes like knot tying and assembly, many shapes satisfy the conditions for a successful task completion. Selecting a single configuration restricts the solution space in these problems.

One of the important application of DLO manipulation planning is automated knot tying for robotic surgery and some assembly tasks. The knot is defined by a sequence of crossings of the DLO with itself. A length parameter is defined along the DLO, and the crossings are ordered by the length at which they occur. A crossing is classified in a binary fashion. A knot is defined uniquely by the number and nature of crossings ordered along the DLO length. Works by Morita et al. [34], Wakamatsu et al. [55], Ladd and Kavraki [28], Saha and Isto [46] and [45], adopted this formulation for defining their planning problems. In [45] Saha and Isto extended this definition to model knot tying around objects in the environment.

Another body of work considered the problem of motion planning for DLOs as a path planning problem for steering a surgical needle inserted in the body to a specific target. In such problems, the steering of needles is treated as an underactuated control problem. Works by Patil et al. [38], [39] and [40], van den Berg et al. [50] and Wen et al. [57], are some of the works that considered planning in this paradigm.

Finally, the goal state of the DLO may also be defined relative to it's environment. Acker et al. [1], [2] describe the system state in terms of contact state of the DLO with respect to the environment. In the goal state, the DLO would need to be in a predefined contact state with the environment. The manipulation actions were

defined as transitions between contact states.

1.1.3 Planning Algorithms

The planning formulations described in the section 1.1.2 lend themselves to various categories of planning algorithms. The geometric or topological definitions grounded in geometry fit into the motion and path planning problem. Works by Moll and Kavraki [33], Bretl [7] and Mukadam et al. [35], propose the use of a probabilistic roadmap planner (PRM) [25] to solve the planning problem with the goal geometry completely specified. The nodes of the planner were sampled from the shape solutions modeled by their respective shape planners. More recently, Roussel et al. [44] combined RRT with sampling from the manifold described by Bretl [7] with a dynamic simulator, in order to account for contact with the environment. They were able to solve routing and positioning problems in highly constrained environments by allowing contact with the obstacles. These technique are useful for planning in instances where the planner must explore configurations with same boundary conditions but very different overall DLO shapes.

Another group of works by Hirai et al. [18], Wada et al. [51] and Berenson [3] proposed reactive control schemes which compute manipulation actions as a function of the error in alignment with respect to the specified final state. Wada et al.[51] modeled the deformable object as a lattice of spring-mass-damper system, similar to the dynamic models and further use a PID control scheme to align the lattice to a reference geometry. Berenson [3] proposed a scheme that approximates the jacobian linking manipulator movement to object movement. This approximate jacobian was then used to drive the pre-defined control points on the object to their commanded location. The advantage of this method is that the jacobian approximation is dependent only on the geometry of the deformable object, and not on physical properties. It only uses a single parameter to model the stiffness of the deformable object. The demerits of this approach are the absence of any theoretical guarantees on task completion or bounds on accuracy of the jacobian.

For the manipulation planning in the knot tying domain, various techniques have

been used to construct a search domain in the various knot configurations. In knot theory, Reidemeister moves are considered as primitive actions for tying knots. Wakamatsu et al.[52] proposed a method that constructs a tree of configurations by applying all possible primitive actions at each configuration. Morita et al. [34] attempted to identify the sequence of reidemeister moves from demonstration by constructing a scheme to identify the move connecting two states, provided they are separated by a single move. Saha and Isto [46] combined the goal definition with a sampling based planner, namely the PRM to rapidly explore their search space.

The contact space domain defined by Acker et al. [1] is more suited to a task planning specification. Each contact state may be modeled as a logical predicate and the actions signifying change of contact states are akin to actions in symbolic planning [5]. Many symbolic task planning techniques exploiting non domain specific heuristics [19] can be used to generate plans in this domain. However the gap between the task level actions and geometric end points for motion planners in this domain is yet to be addressed.

1.2 Thesis Contributions

Most techniques described in the previous section deal with small work pieces which are either grasped at one or both endpoints. Also, the robot manipulating the DLO is assumed to be capable of moving the entire length of the DLO simultaneously. However, when dealing with cables whose lengths are of the same order as lengths of aircraft fuselage, manipulation requires multiple grasp, release and re-grasp actions throughout the length of the cable. This makes it a combined task and motion planning problem. The change of constraints along the cable lengths caused by clamping and grasping constitute the discrete actions. While the actual manipulator positions and cable configurations create the continuous space for motion-level planning. The methods described in section 1.1 are unable to model this problem. Also, the cables in the installation task are interlinked at different points along their length. These interlinks must never be stretched during the installation process, therefore the cables

cannot be treated as independent entities. All previous techniques only model either a single DLO or independently manipulated DLOs.

This thesis provides, to our knowledge, the first planning problem formulation for multiple interlinked DLOs with reconfigurable grasps. The state definitions and the exhaustive set of interaction actions listed in the planning domain set up a rich task planning problem with full geometric information of the system to evaluate feasibility of any action. In this thesis, we also proposes a planner that can generate the task plan to complete the installation process as a sequence of primitive manipulation actions with well defined geometric endpoints. This connects the extensive work done in motion planning algorithms for DLOs with the task level problems at a higher abstraction. Finally, in this thesis we also analyze the optimization problem for computing the shape of the DLO, and propose a property that links geometrically similar shapes across different cable lengths and stiffness values in presence of gravitational forces, allowing for faster shape computation.

Chapter 2

Modeling a Deformable Linear Object

Proactive planning requires the planner to have the capability of predicting the changes to system state occurring due to actions being conducted. In manipulation of deformable objects, a shape computation model allows the planner to predict the shape of the DLO once a manipulation action is executed. The computation of predicted shape allows the planner to detect infeasibilities like a stretched cable or a taut interlink. Thus the ability to predict shape would provide guarantees on feasibility of the plans generated by the algorithm.

Dynamic methods discussed in 1.1.1 can predict the shape of the DLO at all time instances given an initial condition. However, in assembly processes, the movements are slow enough that the DLO being manipulated is in equilibrium, i.e. the shape of the DLO would not change if the manipulator were to stop moving. Hence, computation of dynamic derivatives is unnecessary if the equilibrium shape, given the boundary conditions, can be computed directly. Minimum energy formulations allow just that by framing the problem of shape computation as a curve minimization problem

We adopt an approach similar to that described by Wakamatusu [52], Moll [33] and Bretl [7]. These works state that the cable would assume a shape that minimizes the total elastic energy, which includes a torsional component and a bending component. The minimization problem is subject to endpoint constraints imposed on the cable by either the manipulators, or the objects in the environment like surfaces or clamps.

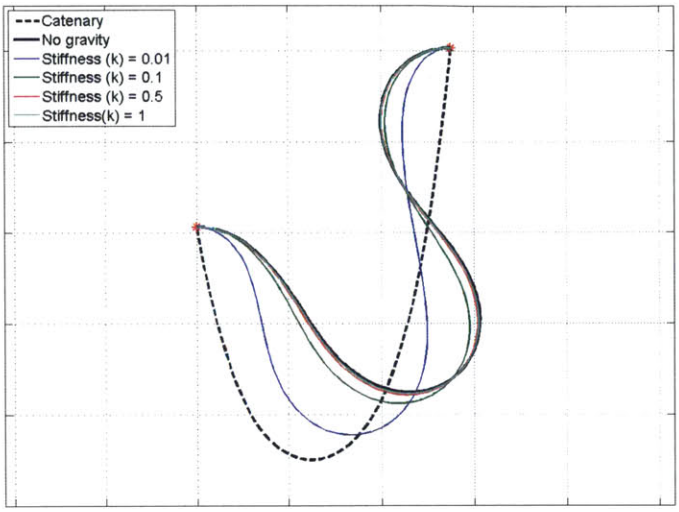


Figure 2-1: Cable shape with identical boundary conditions and varying stiffness

Further, these approaches only model the elastic potential energy of the cable and not the gravitational potential energy. The solutions generated in this case are valid for smaller length and high stiffness DLOs where elastic forces dominate the gravitational forces, but not valid in cases where the DLO length, and hence mass, is large enough to have a significant effect on the shape of the DLO. Figure 2-1 shows the case where gravity has a significant effect on the shape of the cable. The shape of a cable with no elasticity can be modeled as a catenary, as depicted by the dotted black line. The shape of a cable in absence of gravity is depicted by the solid black line. Between these extremes lie a family of curves depending on the relative strengths of the stiffness and gravity.

In section 2.1 we begin by describing the curve optimization problem at the heart of the shape computation problem. We incorporate the gravitational potential energy in the objective functional and define the stiffness parameters proportional to the torsional and bending stiffness of the DLO. In section 2.2 we present and the transformations under which the solutions are invariant or geometrically similar. In section 2.3 we describe the overall shape computation scheme that exploits the invariance and similarity transforms shown in section 2.2. In addition we extend this scheme to a multi point boundary conditions problem where the cable positions are constrained at multiple pre-determined points along the DLO. This shape computation scheme

can be used to predict cable shapes in the installation process described in chapter 3.

2.1 Problem Formulation

A DLO may be defined mathematically as a curve through 3D space parameterized along the length of the DLO. Each point on the DLO must map to a position and an attitude to completely define the cable configuration. Thus a DLO can be represented by the following curve:

$$\mathbf{C} \ : s \rightarrow \mathbb{R}^3 \times S^3; \ s \in [0,1] \tag{2.1}$$

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

The curve may be partitioned into the position and the attitude components as follows:

$$\mathbf{C}(s) = \begin{bmatrix} \mathbf{X}(s)^T & \mathbf{\Phi}(s)^T \end{bmatrix}^T \tag{2.2}$$

where s is a parameter along the length of the DLO. $s = 0$ represents the starting end of the DLO, and $s = 1$ represents the terminal end of the DLO. $\mathbb{R}^3$ represents the position in a 3D euclidean space and S^3 represents the attitude of the frame attached to the cable at the specified point. Here we choose to represent the attitude in terms of the euler angles.

Consider a DLO which is held at both ends in pre-specified configurations given by $\mathbf{C_0}$ and $\mathbf{C_f} \in \mathbb{R}^3 \times S^3$ respectively. The curve $\mathbf{C}$ would describe the configuration of the DLO if it is a solution to the following curve optimisation problem.

$$Localmin \ J = \int_0^1 k_1 \frac{u^2}{L^2} + k_2 \frac{\tau^2}{L^2} + LW_g yds \tag{2.3}$$

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 $\mathbf{C}(1) = \mathbf{C_f}$

where k_1 and k_2 are proportional to the bending and torsional stiffness of the cable, respectively; W_g is the gravitational potential energy per unit length per unit displacement; and u/L and τ/L are the curvature and the torsion in the cable, respectively. Note that as the length along the DLO is parameterized by $s \in [0, 1]$, the scaling factors are applied to the energy terms.

In the governing differential equations, note that:

$$\left(\frac{dx}{ds} + \frac{dy}{ds} + \frac{dz}{ds}\right)^2 = L \tag{2.4}$$

Thus the length of any curve formed by integrating the differential equations from $s = 0$ to $s = 1$ will have the length L. Such curves are called constant speed curves. This ensures that the length of the solution generated will always be constant.

This formulation states that the shape assumed by the DLO must lie in the family of curves that satisfy the end point constraints and have a curve length of L. In addition the curve must be a local minima of the functional J. The functional J may have multiple local minima with the same end point constraints and the particular shape assumed by a physical DLO depends on the path taken by its end points to reach the specified position. In context of the assembly process, we assume that the shape of the DLO will be the one that does not have any loops.

As stated, the curve optimization problem is mathematically identical to optimal control problems. The necessary conditions that a local minima of the curve optimization problem must satisfy a two point boundary value problem (BVP). The approach taken to derive the BVP is well studied in literature [27]. The necessary

conditions that a local minima for the problem 2.3 must satisfy can be stated as the following boundary value problem.

$$\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{dz}{ds} \\ \frac{d\phi}{ds} \\ \frac{d\theta}{ds} \\ \frac{d\psi}{ds} \\ \frac{d\lambda_1}{ds} \\ \frac{d\lambda_2}{ds} \\ \frac{d\lambda_3}{ds} \\ \frac{d\lambda_4}{ds} \\ \frac{d\lambda_5}{ds} \\ \frac{d\lambda_6}{ds} \end{bmatrix} = \begin{bmatrix} L\ cos(\theta)cos(\psi) \\ L\ cos(\theta)sin(\psi) \\ L\ sin(\theta) \\ \tau + u\ tan(\theta)sin(\phi) \\ u\ cos(\phi) \\ u\ sec(\theta)sin(\phi) \\ 0 \\ -L\ W_g \\ 0 \\ L\ f_1 + L^2\ f_1' \\ L\ f_2 + L^2\ f_2' \\ L\ f_3 + L^2\ f_3' \end{bmatrix} \tag{2.5}$$

where

$$u = \frac{-L^2}{k_1} \sum_{i=4}^{6} \lambda_i\ g_{1_i}(\theta, \phi, \psi)$$

$$\tau = \frac{-L^2}{k_2} \sum_{i=4}^{6} \lambda_i\ g_{2_i}(\theta, \phi, \psi)$$

$$f_k = \sum_{i=1}^{3} \lambda_i\ f_{k_i}(\phi, \theta, \psi); k \in \{1, 2, 3\}$$

$$f_k' = \sum_{i=4}^{6} \sum_{j=4}^{6} \lambda_i\ \lambda_j\ f_{,k_{ij}}(\phi, \theta, \psi, k_1, k_2); k \in \{1, 2, 3\}$$

subject to

$$\mathbf{C}(0) = \mathbf{C_0} \text{ and } \mathbf{C}(1) = \mathbf{C_f}$$

In the case that only the initial end of the DLO is constrained and the terminal end of the DLO is free, the boundary conditions are as follows:

$$\mathbf{C}(0) = \mathbf{C_0} \; and \; \boldsymbol{\lambda}(1) = \mathbf{0} \tag{2.6}$$

The detailed derivation of this boundary value problem is provided in appendix A. Two-point boundary value problems such as these can be solved using numerical collocation schemes. We implement the shape computation scheme in MATLAB using the inbuilt implementation of the Lobatto IIIA scheme in the function 'bvp4c' [**26**].

2.2 Properties of The Solution Curve

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 a cable with length L and stiffness values k_1 and k_2 is geometrically similar to that for a cable with 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 and the mass per unit length of the DLO is constant.

The first two of these properties are intuitive as the shape of a physical DLO does not change when subjected to a pure translation, or a rotation along the axis aligned with the gravity vector. These transforms can be exploited by designing a scheme placing the origin of the coordinate system at the initial end of the DLO. The solution can then be translated and rotated as required to align it with the respective boundary conditions.

The third property defines the relationship between the stiffness parameters and the cable lengths for a given linear mass density that allows simple geometric scaling. It states that a DLO with a given stiffness k, and length L will behave like a DLO of

a shorter length L' with lesser stiffness, more specifically scaled by the third power of the lengths, i.e. $\frac{kL'^3}{L^3}$. A proof for this property is provided in appendix B.

The evaluation of the necessary conditions for a local minima to the functional J defined in equation A.1 requires solving a two point boundary value problem. Given that the equations describing the BVP are non-linear and non-convex, the numerical solution methods are not guaranteed to converge to a solution. The convergence is dependent on the closeness of the inital guess provided to the solver to the actual solution. It was empirically seen that that a solution to the curve optimization problem for a given DLO length, stiffness and boundary condition can act as a good initial guess for the optimization problem for a DLO with other stiffness parameters and boundary conditions, but with an identical length. Using the solution as an initial guess for a problem with different DLO length would result in a failure to converge to the solution.

The properties proved in this section allow for design of a computation scheme where the BVP must only be solved for problems involving the same length, with one of the end points being at the origin. The invariance and similarity properties can then be used to scale the solution to the required length, and align it to the required boundary conditions.

2.3 Overall Shape Computation Scheme

With the curve optimization formulation of the shape computation problem defined in section 2.1, combined with the properties proved in section 2.2, a scheme to compute the shape of a cable is defined in this section. First, a case of a DLO with only its endpoints constrained is discussed to demonstrate the use of transforms to compute the DLO shape given one known solution for a specific DLO length, stiffness and boundary conditons. Next, the scheme is generalised to a DLO with constraints at multiple known points along its length.

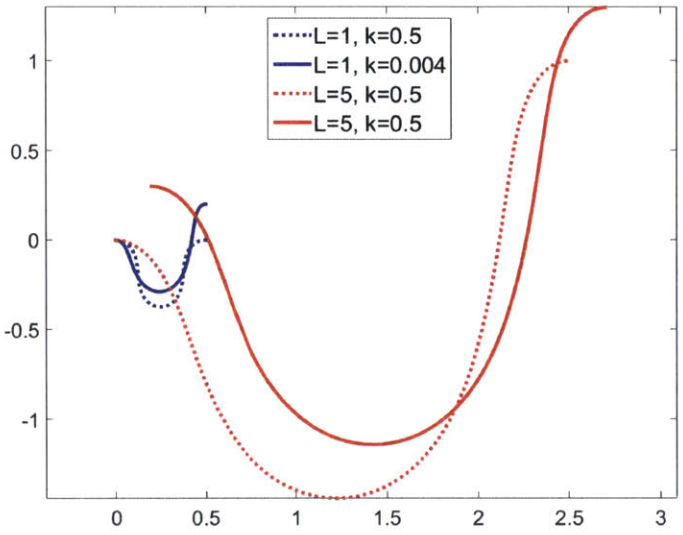


Figure 2-2: Example of steps followed for shape computation.

2.3.1 Two-Point Boundary Conditions

Consider the case where a the shape of a DLO constrained to be in the XY plane must be predicted. In this case, the curve that describes the configuration can be defined as $\mathbf{C}(s) = \begin{bmatrix} x(s) & y(s) & \theta(s) \end{bmatrix}^T$. The DLO has a length of $L = 5m$ and with stiffness parameter $k_1 = 0.5$. The endpoints are constrained at $\mathbf{C}(0) = \begin{bmatrix} 0.2 & 0.3 & 0 \end{bmatrix}$ and $\mathbf{C}(1) = \begin{bmatrix} 2.7 & 1.3 & 0 \end{bmatrix}$. The solution $\mathbf{C}'(s)$ for the problem with length $L' = 1\ m$ and stiffness $k_1' = 0.5$ and boundary conditions $\mathbf{C}'(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and $\mathbf{C}'(1) = \begin{bmatrix} 0.5 & 0 & 0 \end{bmatrix}^T$ is known (shown by the dotted blue line in figure 2-2).

The steps taken by the computation scheme to compute the shape of the specified problem given the initial solution are as follows:

1. Provide the known solution as an initial guess to a boundary value problem solver and compute the shape $\mathbf{C}(s)_{temp}$ for $L'_{temp} = 1$ m and $k_{1_{temp}} = k_1' \frac{L'^3}{L^3} = 0.004$ with boundary conditions $\mathbf{C}(0)_{temp} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and $\mathbf{C}(1)_{temp} = \begin{bmatrix} \frac{[\mathbf{X}(1) - \mathbf{X}(0)]^T}{L} & \mathbf{\Phi}(1) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.2 & 0 \end{bmatrix}^T$ (See the solid blue line in Figure 2-2).

2. Scale $\mathbf{C}'_{temp}(s)$ by a factor of 5 (See the dotted red in Figure 2-2). This generates the shape for a DLO of length $L = 5$ m and stiffness $K_1 = 0.5$ according to property 3.

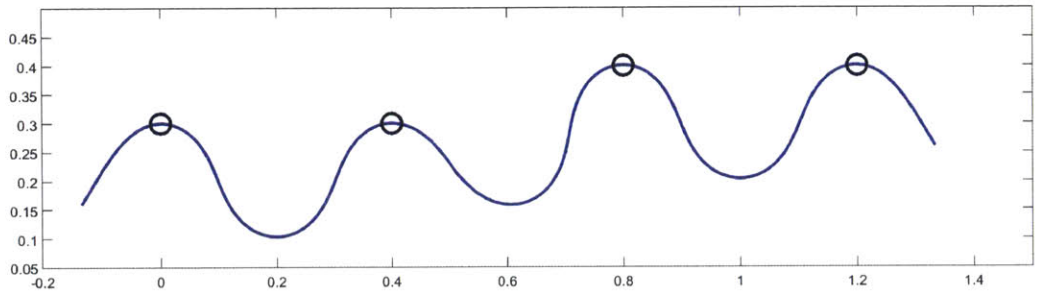


Figure 2-3: Shape computation for a multi-point boundary value problem.

3. Translate the positions of $\mathbf{C}'(s)$ by $\begin{bmatrix} 0.2 & 0.3 \end{bmatrix}^T$ to align the DLO with the correct initial position. (See the solid red line in Figure 2-2).

2.3.2 Multi-Point Boundary Conditions

If a DLO has constraints defining the configuration at n different pre-specified points along the length of the cable, then the shape of the entire cable can be computed by sequentially solving $n+1$ two point BVPs. For example, consider a DLO of total length $L_{tot} = 2.2\ m$, with its shape represented by the curve $\mathbf{C}(s)$. It is attached to clamps that fix the configuration at 4 points with the following boundary conditions:

$$\begin{aligned} \mathbf{C}(0.091) &= \begin{bmatrix} 0 & 0.3 & 0 \end{bmatrix}^T \\ \mathbf{C}(0.367) &= \begin{bmatrix} 0.4 & 0.3 & 0 \end{bmatrix}^T \\ \mathbf{C}(0.636) &= \begin{bmatrix} 0.8 & 0.4 & 0 \end{bmatrix}^T \\ \mathbf{C}(0.909) &= \begin{bmatrix} 1.2 & 0.4 & 0 \end{bmatrix}^T \end{aligned} \tag{2.7}$$

The overall shape is computed by computing the individual boundary value problems followed by concatenating the solutions. The individual problems solved are as follows:

1. A free-fixed problem between $s = 0$ and $s = 0.091$.

2. A fixed-fixed problem between $s = 0.091$ and $s = 0.367$.

3. A fixed-fixed problem between $s = 0.367$ and $s = 0.636$.

4. A fixed-fixed problem between $s = 0.636$ and $s = 0.909$.

5. A fixed-free problem between $s = 0.909$ and $s = 0.1$.

Chapter 3

Planning Domain

In this chapter, we provide a mathematical definition of the manipulation planning problem underlying the cable installation process. A planning problem formulation must include a representation of the various objects involved in the process to define the state of the system at a given time instant. Additionally, actions are encoded as functions that effect a change of the state variables. Finally, tests must be defined to identify the feasible states that the process may enter into and infeasible states which the planner must avoid. To that end, the proposed formulation includes mathematical representations for the system state, interlink constraint, primitive action types, a structure for a task plan and a validation test for a given task plan.

3.1 Task Components

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

1. **Cables:** Each cable in the planning problem is defined by the following components:

   (a) **Cable shape**: The shape of all cables involved in installation process is maintained as a part of the state at each time instant. Each cable's shape

is represented by a curve parameterized 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, $\mathbf{C^i}$, as follows:

$$\mathbf{C^i}: \ s \rightarrow \mathbb{R}^3 \times S^3 \tag{3.1}$$

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

$\mathbb{R}^3$ is the three dimensional euclidean space used to encode the position of the point of the cable. S^3 is the hypersphere embedded in four dimension, used to encode the attitude of the point on the cable. In this thesis, we use the Euler angle parameterization for the attitude. L^i is the length of the i^{th} cable. Additionally the position and orientation are partitioned 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 configuration of the cable at the specified length. The clamp positions for cable i are denoted by an ordered array of K^i as follows:

$$\mathbf{K^i} = \left[K^i_j\right]; \ j \in \{1, ...n^i_{clamps}\}; \ K^i_j \in \mathbb{R}^3 \times S^3 \tag{3.2}$$

An ordered pair $\left(K^i_j, s^i_j\right)$ where $s \in [0, L^i]$; denotes that the point s^i_j on the cable i is fixed to the clamp position K^i_j. In this instance, the configuration of the cable at s^i_j is fixed with respect to the clamp position as follows

$$\mathbf{C^i}(s) = f_K(K^i_j) \tag{3.3}$$

Where $f_K : \mathbb{R}^3 \times S^3 \rightarrow \mathbb{R}^3 \times S^3$ is an invertible function that maps the clamp position to a configuration of the cable held by the clamp. The function

f depends on the geometry of the clamps being used to hold the cable. To attach the cable at clamp position K_j^i, the cable must be positioned at $\mathbf{C^i}(s_j^i) = f_K^{-1}(K_j^i)$. In this thesis, a simple mapping $f(K_j^i) = K_j^i$ is used.

(c) **Reference points**: Reference points are locations on the cable that must, ideally, be used to clamp the cable in the final position. The number of reference points is equal to the number of clamps, and each reference point is linked to a corresponding clamp. The reference points for the i^{th} cable are denoted by an ordered array as follows:

$$\mathbf{R^i} = \left[r_j^i\right]; j \in \{1, 2, ...n_{ref}^i\}; r_j^i \in \left[0, L^i\right] \tag{3.4}$$

Each reference point is parameterized by its position along the length of the cable.

(d) **Gripping Points**: The electrical cables may often have electrical component protruding out at certain positions. Clamping the cables, or grasping them tightly with a manipulator, at these points, may result in damage to the cables. Hence a set of discrete "gripping" points are defined for each cable, where they can be safely grasped or clamped. A cable can only be gripped by manipulators at these gripping points. For the i^{th} cable, they are defined as an ordered array $\mathbf{G^i}$ as follows:

$$\mathbf{G^i} = \left[g_j^i\right]; j \in \{1, 2...n_{grip}^i\}; g_j^i \in \left[0, L^i\right] \tag{3.5}$$

The position of the gripping points is parameterized by their position along the length of the cable.

2. **Manipulators**: Each manipulator configuration is defined by it's position and orientation. In this formulation it is assumed that the manipulators are unrestricted in their movements. The manipulators are part of the set $\mathbf{M}$:

$$\mathbf{M} = \{M_k\}; k \in \{1, 2, ...n_{manipulator}\} \tag{3.6}$$

$$M_k \in \mathbb{R}^3 \times S^3$$

A manipulator can either be "free" or "occupied". An occupied manipulator is described by the ordered pair (M_k, g_j^i), implying that the k^{th} manipulator is grasping cable i at grip point g_j^i. The configuration of the cable at g_j^i is fixed by the manipulator through the following function

$$\mathbf{C}^i(g_j^i) = f_M(M_k) \tag{3.7}$$

$g : \mathbb{R}^3 \times S^3 \rightarrow \mathbb{R}^3 \times S^3$ is an invertible function that maps the transformation from manipulator position to the position it would hold the cable at, and depends on the geometry of the manipulator used. To grasp a manipulator at a given gripping point g_j^i, the manipulator must position itself at $M_k = f_M^{-1}(\mathbf{C^i} g_j^i)$. In this thesis, a simple mapping $f_M(M_k) = M_k$ is used.

3. **Interlink constraints**: Interlinks are modeled as geometric constraints, which must be satisfied in a valid system state. All interlinks are contained within a set **I**, with each element is described by a 5-tuple as follows:

$$\mathbf{I} = \{(i_k, j_k, s_k^i, s_k^j, l_k)\}; k \in \{1, 2, ...n_{links}\} \tag{3.8}$$

$$i_k, j_k \in \{1, 2, ...n_{cable}\}; s_k^i \in [0, L^i]; s_k^j \in [0, L^j]; l_k \in \mathbb{R}^+$$

Here, i_k and j_k are the indices of the cables involved in interlink k, s_k^i and s_k^j 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 following condition holds true:

$$||\mathbf{X^{i_k}}(s_k^i) - \mathbf{X^{j_k}}(\mathbf{s_k^j})|| < l_k \tag{3.9}$$

Equation 3.9 states that the interlink constraint is satisfied if the distance between the attachment points is less than the length of the interlink. If the distance

between the attachment points exceeds l_k, then the interlinks is guaranteed to be taut, conversely, if the distance between the attachment points is less than l_k, then the interlinks is guaranteed to be slack. Due to the relative sizes of the interlink and the main cables, the interlinks are assumed to not affect the shape of the cables.

3.2 System State

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

$$\mathbf{S} = \left(\{\mathbf{C^i}\}, \{(K_j^i, s_j^i)\}, \mathbf{M}, \{(M_k, g_j^i, \mathbf{F})\}\right) \tag{3.10}$$

The elements of the tuple are as follows:

1. $\{\mathbf{C^i}\}$, is the set of curves describing the cable shapes.
2. $\{(K_j^i, s_j^i)\}$, 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. $(\{M_k, g_j^i)\}$, 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 (3.9)

Figure 3-1 depicts an example system state. The red lines indicate taut interlinks and the green lines indicate slack interlinks that satisfy the constraint in equation (3.9).

3.3 State Transition Actions

The state transition actions are the set of actions that affect the state of the system. For the scope of this work we assume that a single manipulator can only grasp one

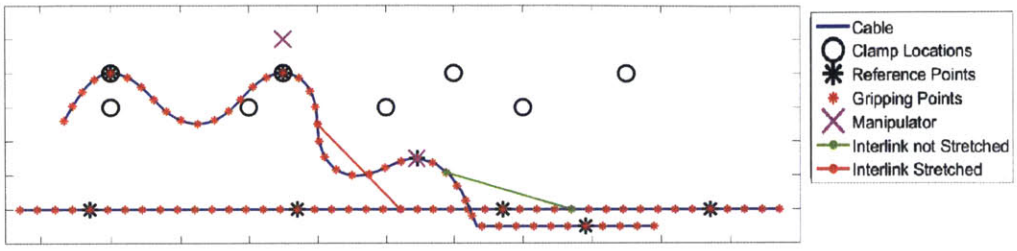
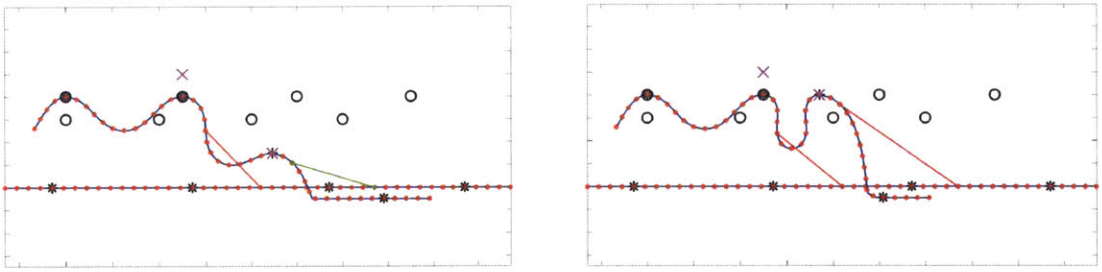


Figure 3-1: A visual representation of the System State

cable at a time, and the only modes of change to the system state are through the manipulators performing a sequence of the primitive actions. We define a set of four primitive actions and one compound action that exhaustively model the interactions of the manipulators with the cables. Each action can effect a change in the system state by either changing the boundary conditions describing the shapes, attachments of the cable to the clamps, or the gripping conditions between the cable and the manipulators. At the end of each primitive action, the shapes of all the cables are recomputed and the satisfaction of each interlink constraint is evaluated.

1. **RepositionManipulator**(`ManipID`, `target`)

   This action type respositions the selected manipulator to the specified target. `ManipID` $\in \mathbf{M}$ and `target` $\in \mathbb{R}^3 \times S^3$. An example of the state before and after a RepositionManipulator actions is carried out is depicted in figure 3-2.



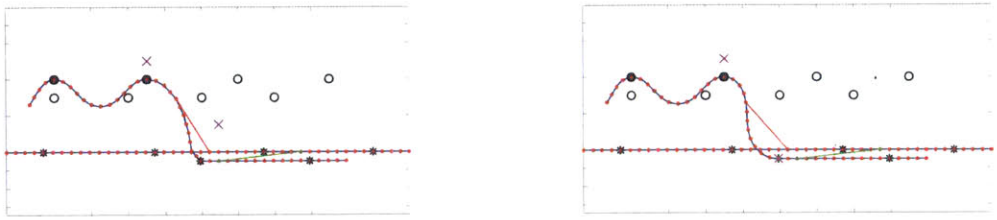
(a) (b)

Figure 3-2: Example pre-conditions and post-conditions for RepositionManipulator action.

2. **GraspCable**(`ManipID`, `CableID`, `GripPointID`)

   This action results in the selected manipulator moving and grasping the spec-

ified cable at the specified gripping point. `ManipID` $\in \mathbf{M}$, `CableID` $= i \in \{1, 2, ...n_{cable}\}$, `GripPointID` $\in \mathbf{G^i}$. An example of the state before and after a GraspCable action is carried out is depicted in figure 3-3.

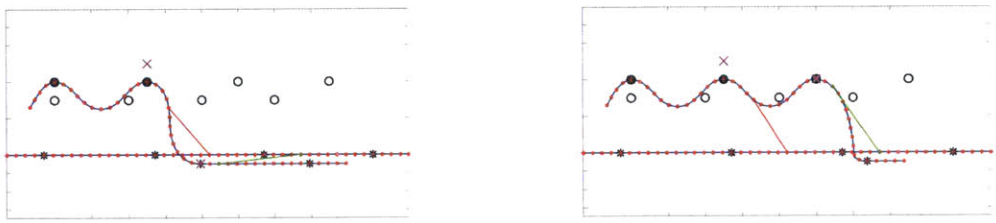


(a) (b)

Figure 3-3: Example pre-conditions and post-conditions for GraspCable action.

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` $\in \mathbf{K^i}$. An example of the state before and after a GraspCable action is carried out is depicted in figure 3-4.

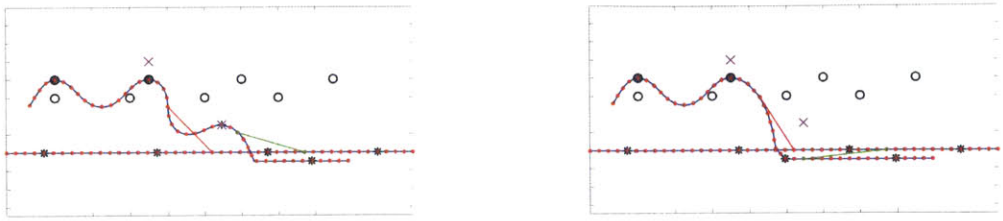


(a) (b)

Figure 3-4: Example pre-conditions and post-conditions for ClampCable action.

4. **ReleaseManipulator**(`ManipID`)

   This action frees the selected manipulator and removes the boundary condition from the grasped cable. `ManipID` $\in \mathbf{M}$. An example of the state before and after a GraspCable action is carried out is depicted in figure 3-5.



(a) (b)

Figure 3-5: Example pre-conditions and post-conditions for ReleaseManipulator action.

5. **AlignReferencePoint**(`ManipID`, `CableID`, `RefID`)

   This is a compound action composed of two primitive actions. 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.

3.4 Plan Structure

The manipulation plan is described as the following tuple:

$$\mathbf{P} = \left(\{A_1^1, A_2^1, \ldots\}, \{A_1^2, \ldots\}, \ldots, \{A_1^k, \ldots\}, \ldots \{A_1^n, \ldots\}\right) \tag{3.11}$$

Equation (3.11) 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. A visual depiction of a manipulation planning showing the simultaneous primitive actions and the temporally disjoint *action sets* is depicted in figure 3-6.

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

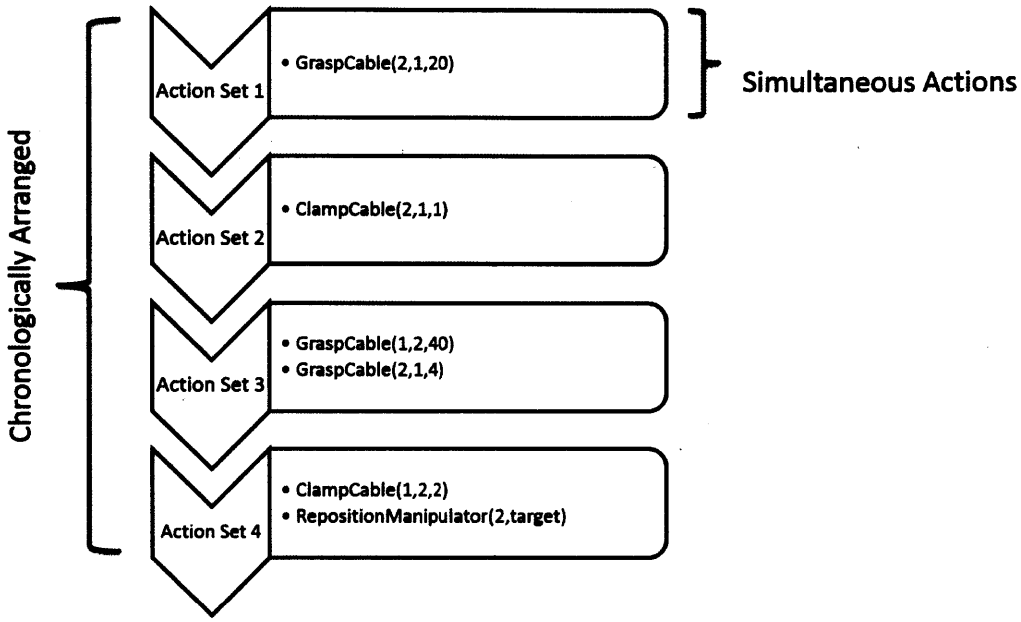


Figure 3-6: A visual representation of an example manipulation plan

action set are completed.

Chapter 4

Planning Algorithm

As seen in chapter 3, the planning domain relies on a mixed continuous and discrete space definition of the system state. The manipulation actions, defined as an exhaustive set in section 3.3, change the state by either changing the boundary condition, reconfiguring the clamp and manipulator usage and changing the set of violated interlink constraints. Together the planning domain would closely match the classical predicate based domain akin to PDDL [5], [19] based domain definitions. However PDDL-like planning domains operate exculsively on logical predicates with actions that have either deterministic of probabilistic effect on predicates. An extension, PDDL+ [14], considers continuous processes too, but with effects of the action known a priori. The planner domain, defined in this thesis, has predicates whose truth values are dependent on continuous arguments in the actions. Thus, classical PDDL-like or even PDDL+ definitions cannot capture the planning domain adequately. As a result, the vast array of planning techniques developed for these languages cannot be used for manipulation planning problems. In recent work Srivastava et al. [48], Garrett et al. [15] and Lozano Perez and Kaebling [31] attempted to combine task-level planning with motion-level planning, but they are valid only for the pick-and-place task domain. Hence, we propose a novel planning algorithm to tackle problems generated in the formulation described in chapter 3.

The prototype planner proposed here accepts, as input, a list of reference points to be aligned, $\{(i, r_j^i)\}$; where $i \in \{1, \ldots, n_{cable}\}$; and $r_j^i \in \mathbf{R^i}$; and the initial state

of the system **S** as defined by equation (3.10). The output of the planner is a valid task plan **P** in the format defined in (3.11).

The proposed algorithm attempts alignment of reference points in the order provided as input. If the planner fails to align the cable using the compound action **AlignReferencePoint**, defined in section 3.3, then it utilizes a set of pre-defined resolution strategies to generate the plan for resolving interlink constraint violations. The planner exits when all the reference points have been aligned or it reaches a state where it cannot plan for alignments of any of the remaining reference points.

4.1 Planner in a nutshell

The first algorithm to be called is the **AlignReferencePointList** (algorithm 1), with input arguments including the list $\{(i, r_j^i)\}$; the initial state **S**; an initial plan **P** which begins as an empty set; the list of free manipulators, which initially consists of the entire set of manipulators; and the occupied manipulators list which begins as an empty set. This algorithm attempts to align the reference points in $\{(i, r_j^i)\}$ in their listed order. An alignment is identified as successful if and only if a plan can be found such that the specified reference point is aligned with it's clamp and all the manipulators included in the free manipulators list at input are free upon completion of its execution. If the alignment fails, the algorithm attempts to align the next point in the list. Algorithm 1 exits successfully if all reference points in $\{(i, r_j^i)\}$ are aligned, and exits with a failure if the algorithm revisits a reference point, for which alignment has already failed, without performing any successful alignments in between attempts.

If a simple alignment using **AlignReferencePoint** is not possible for the queried reference point without violating an interlink constraint, **ResolveInterlinkConflict** (algorithm 2) is called. This algorithm attempts to use the two resolution strategies implemented by the planner. It first attempts a single step resolution where additional free manipulators are used to simultaneously align reference points on other cables, with clamps close to the reference point that was initially being aligned. If single-step resolution fails for some cables, the algorithm then attempts a repositioning

based resolution by calling **AttemptRepositionResolution** (algorithm 3). Here, the planner selects a set of grip points on the corresponding cables, and uses the manipulators to position the cables such that no interlinks are taut. The manipulator and cable positions are computed by **ComputeGeometricResolutions** (algorithm 5).

The use of repositioning-based resolution can result in some of the initial free manipulators being occupied. A successful reference point alignment requires that all initially free manipulators are free upon completion of the alignment. To achieve this, additional reference points need to be aligned to free up the occupied manipulators. **DetermineRefPointsToAlign** (algorithm 4) determines the set of additional reference points that need to be aligned for successful resolution. **AttemptRepositionResolution** then recursively calls **AlignReferencePointList** with the new set of reference points, and free and occupied manipulator list at that state. If a plan to align the subset of reference points can be found, the resolution and alignment are both successful and the planner can move to the next reference point in the original list.

4.2 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 the input state to the 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 FailList, a list of elements of RefPointList with failed alignment attempts. Line 7 selects an unaligned reference point. In lines 9 and 10, the algorithm exits with a failure if attempted alignment of the selected reference point fails twice

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 ← Select Manipulator from FreeManipulators
7:         (i, r_j^i) ← Select from RemainingRefPointList
8:         Delete (i, r_j^i) from RemainingRefPointList
9:         if (i, r_j^i) ∈ FailList then
10:            exitFlag = True; SuccessFlag = False; Break
11:        else
12:            S.AlignReferencePoint(M,i,r_j^i)
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_j^i) 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
```

with no successful alignment in between the two attempts. 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 **P**. In the event of violated interlink constraints, line 19 attempts resolution using algorithm 2. If this 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 **S** to the state prior to the attempted alignment, and add the selected reference point to the end of the queue. Line 27 declares success and exits if all reference points are aligned.

4.3 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, $\{(M, g_j^i)\}$, where $i \in \{1 \dots n_{cable}\}$, $M \in \mathbf{M}$, $g_j^i \in \mathbf{G^i}$; 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 these conditions are not met, 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, $\mathbf{k}$ 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 4-1 (a) depicts an instance of a taut interlink and reference points in $\{r\}$. All reference points between R1 and R2 (two in the case presented) are included

Algorithm 2 ResolveInterlinkConflict

1: **function** RESOLVEINTERLINKCONFLICT (**S**, **P**, FreeManipulator, OccupiedManipulators, $(i_{clamped}, M_{clamped})$)
2: Initialise CorrespondingCables
3: **if** size(CorrependingCables) - size(OccupieManipulators) > 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 $(M, g_j^i) \in$ 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 $(M, g_j^i) \notin$ 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

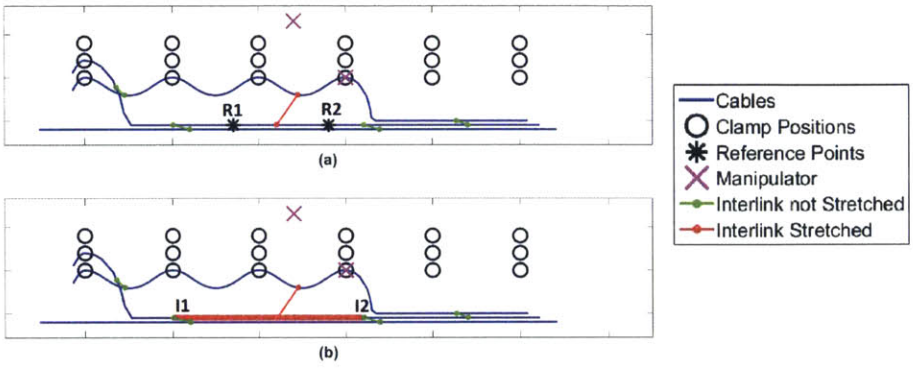


Figure 4-1: An instance of a violated interlink and the considered reference and gripping points.

in the set $\{r\}$. $\{g\}$ is a set of grip points that will be considered for grasping the cable in order to reposition it. As shown in Figure 4-1 (b) all gripping points between the first slack interlinks to the left (I1) and right (I2) 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 corresponding cables that are not grasped. 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. 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 just the single-step resolution strategy. Line 19 uses algorithm 3 to plan a repositioning based resolution strategy. Lines 20 to 27 exits the algorithm with a flag indicating success or failure of the repositioning attempt.

Next, we consider the function **AttemptRepositionResolution** (algorithm 3). This function is called in algorithm 2 in the event that single-step resolution fails. It accepts as input the system state $\mathbf{S}$, the task plan $\mathbf{P}$ and the CorrespondingCables-Manip set of tuples from algorithm 2. It also requires RepositionList $\{i\}$, a list of cables that require repositioning as determined in algorithm 2, where $i \in \{1 \ldots n_{cable}\}$.

Algorithm 3 AttemptRepositionResolution

1: **function** ATTEMPTREPOSITIONRESOLUTION(**S**, **P**, RepositionList, CorrespondingCablesManip)
2: **if** $\exists\, m$ 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 $\geq$ 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: $g_j^i \leftarrow$ Sample grip point from $\{g\}$
12: **S**.GraspCableParallel(m, i, g_j^i)
13: Delete g_j^i from $\{g\}$
14: **else**
15: Increment GripListEmptyCount
16: (**S**, **P**, RepositionFlag) = ComputeRepositionResolution(**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

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 upon input to algorithm 2 are released.

From lines 2 to 3 the additional set of reference points requiring alignment are determined using algorithm 4. For each free manipulator, there exists a list of grip points that must be attempted in order 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 assigned free manipulator, in the ascending order according to the priority function defined in equation (4.1). 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 at line 3 using algorithm 1. Finally, lines 22 to 25 declare success and exit. (The algorithm continues if the alignment of the reference point list fails.)

Algorithm 4 DetermineRefPointsToAlign

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

Next, we discuss the function **DetermineRefPointsToAlign**(algorithm 4). This algorithm takes as input the system state **S** after a reference point alignment, the set

of tuples CorrespondingCablesManip and the RepositionCables list from algorithm 2. Also, as the function has been designed as a recursive algorithm, it accepts an initial alignment list of reference points: $\{(i, r_j^i)_m\}$, where $i \in \{1, \ldots, n_{cable}\}; r_j^i \in \mathbf{R^i}$. The reference point alignments in this function exist only in planner memory and 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. Relevant reference points are aligned in ascending order of the priority function evaluated for $s = r_j^i \; \forall \; r_j^i \in \mathbf{k}$:

$$f(s) = \sum_{(i,a,s^i,s^a,l)\in \mathbf{j}} |s^i - s|; s \in [0, L^i] \tag{4.1}$$

The priority function is the summation of the absolute distance of the selected reference point from all taut interlinks along the length of the cable. This function 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 at line 15 while providing the alignment list determined thus far as an input.

4.4 Computing a Resolution to the Interlink Conflict

In the event that the attempted 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 the continuous space of manipulator positions. In order to address this problem, we use a technique similar to the that described by Berenson [3].

Consider a cable j being repositioned using manipulator l with grip point m at

length g_m^j. 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 $F(M_l) : \mathbb{R}^3 \times S^3 \rightarrow \mathbb{R}^{3P}$ map the manipulator configuration to the positions of the points in $\mathcal{P}$ as follows:

$$\begin{aligned} \mathcal{P} &= F(M_l) \\ \dot{\mathcal{P}} &= \frac{\partial F}{\partial M_l}\dot{M}_l = J\dot{M}_l \end{aligned} \tag{4.2}$$

Berenson [3] used 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" as follows:

$$\mathbf{J_i} = w_i \mathbf{J_{i_{rigid}}}; i \in \{1, 2, \ldots P\} \tag{4.3}$$

$$\mathbf{J} = \begin{bmatrix} \mathbf{J_1}^T & \mathbf{J_2}^T & \ldots & \mathbf{J_i}^T & \ldots & \mathbf{J_P}^T \end{bmatrix}^T \tag{4.4}$$

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

$$w_i = e^{-k|g_m^j - s_{p_i}^j|} \tag{4.5}$$

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 defined as follows:

$$\mathbf{J_{i_{rigid}}} = \begin{bmatrix} \mathbf{I_3} & \mathbf{J_{i_{rot}}} \end{bmatrix} \tag{4.6}$$

$$J_{i_{rot}} = [r]_{\times}^{T} \mathbf{T_{IM}} \mathbf{A} \tag{4.7}$$

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 follows:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -sin\theta \\ 0 & cos\phi & sin\phi cos\theta \\ 0 & -sin\phi & cos\phi cos\theta \end{bmatrix} \tag{4.8}$$

The control input applied to the manipulator are computed as follows:

$$\dot{M}_l = \mathbf{J}^{\dagger}(\mathcal{T} - \mathcal{P}) \tag{4.9}$$

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

The complete method 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. CorrespondingCablesManip and RepositionCables from algorithm 2 must also be provided as input. If the algorithm is 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.

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 4.9 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 all interlink constraints are satisfied.

Algorithm 5 ComputeGeometricResolutions

```
1: function ComputeGeometricResolution (S, P, CorrespondingCablesManip, Reposition-
   Cables)
2:     for all (i, k, {r}, {g}, M) ∈ CorrespondingCablesManip such that i ∈ RepositionList do
3:         Initialize 𝒫 and 𝒯 vectors
4:     while exitFlag == false or iterCount ≤ MaxIter do
5:         for all (i, k, {r}, {g}, M) ∈ CorrespondingCablesManip such that i ∈ RepositionList do
6:             if Interlinks on i are taut then
7:                 Add attachment points of new taut interlinks to 𝒫 and 𝒯
8:                 J ← Compute Jacobian
9:                 δM_k ← Propagate control
10:                S.RepositionManipulator(k, M_k + δM_k)
11:            if iterCount ≥ 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
```

Chapter 5

Applications and Analysis of the Planning Algorithm

In this chapter, we analyze the proposed algorithm by studying an instance of its execution on a small scale problem designed to test all the capabilities of the planning algorithm. That is followed by derivation of the worst case time complexity of the algorithm and a discussion of properties required for a complete algorithm. Finally, we demonstrate the use of the planning algorithm on a real-world cable installation scenario in an aircraft currently in production.

5.1 Example Solutions

Consider the scenario depicted in figure 5-1. A set of cables $\mathbf{C}^i = \{1, 2, 3\}$, each of length 5.9 m , are laid out on a floor in the initial state. Each cable has a set of clamps $\mathbf{K}^i$ that it must be attached to with clamp positions known a priori. The reference points $\mathbf{R}^i$ are defined along the cable lengths. The grip points $\mathbf{G}^i$ are distributed along the cables at uniform intervals. There are two manipulators $\mathbf{M} = \{1, 2\}$ available in the problem. Finally, there are five interlinks, three between cables 1 and 2 and two between cables 2 and 3. Each reference point is indexed as an ordered pair (i, j), where i is the index of the cable and j is the index of the interlink along that cable. (Below, the cables, manipulators and the reference points are referred to by their

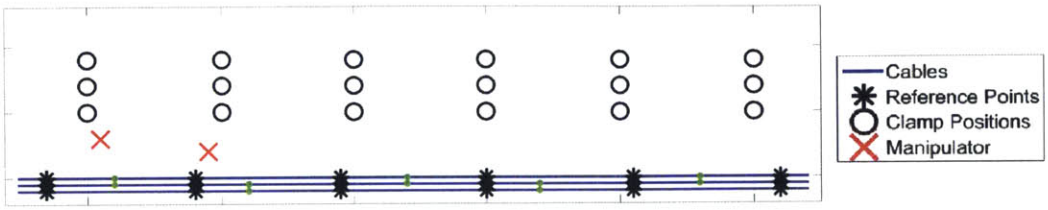


Figure 5-1: Initial State of the problem

respective indices.)

The goal is to align all the reference points on cable 1. The planner is called with **AlignReferencePointList** with the following arguments: The start state **S**; the plan **P**, which is an empty set; the FreeManipulators list, represented by the set $\{1, 2\}$; the list, OccupiedManipulators list, which is an empty set; and the RefPointList represented by the set $\{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5)\}$.

As depicted in figure 5-2 (a), if reference point (1,1) is aligned using manipulator 1 without any additional actions, an interlink between cables 1 and 2 is stretched; hence **ResolveInterlinkConflict** is called with the following arguments: The state **S** of the system; the plan **P** so far; the list FreeManipulators is $\{2\}$; and the OccupiedManipulators list is $\{1\}$. The set of relevant reference points for this problem includes $\{r\} = \{(2, 1), (2, 2)\}$. The free manipulator is assigned to cable 2 to plan for the resolution. The planner first attempts a single step resolution with reference point $(2, 1)$; this attempt is successful, as shown in figure 5-2(b). The reference point $(1, 1)$ is now removed from the RefPointList and the resolution actions are added to **P**.

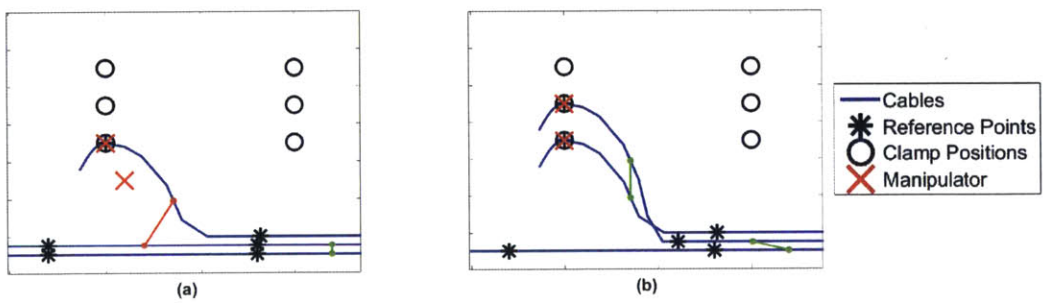


Figure 5-2: Example single step resolution

The planner then attempts to align reference point $(1, 2)$. The alignment is successful with no additional resolution actions required.

Next, the planner attempts to align reference point $(1, 3)$ with manipulator 1. Without resolution actions, this would stretch the interlink between cables 1 and 2; hence, **ResolveInterlinkConflict** is called in order to attempt a resolution. The reference points to be considered for single step resolution are $\{(2, 3), (2, 4)\}$. Aligning either point results in stretching an interlink between cables 2 and 3; therefore repositioning-based resolution is required. It is determined using **DetermineRefPointsToAlign** that reference points $\{\mathbf{r}\} = \{(2, 3), (3, 2)\}$ must be aligned in order to free up the manipulators. Manipulator 2 is then used to grasp cable 2 and reposition it such that the interlink is slack. Next, **AlignReferencePointList** is called with the following arguments: The set FreeManipulators is $\{1\}$; the set OccupiedManipulators is $\{2\}$; the set RefPointList is $\{\mathbf{r}\}$. Aligning reference point $(2, 3)$ is not possible as doing so would stretch the interlink between cables 2 and 3 with no free manipulators to resolve the interlink conflict. As a result, the algorithm attempts alignment of reference point $(3, 2)$. This alignment would also stretch the interlink between cables 2 and 3; however manipulator 2 which has already grasped cable 2 can be used to plan a reposition-based resolution. Once $(3, 2)$ is aligned, the reference point $(2, 3)$ can also be aligned using manipulator 1 without stretching any other interlinks.

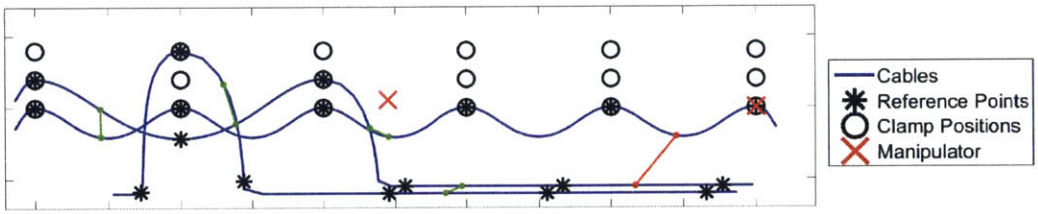


Figure 5-3: An interlink violated during reference point alignment

The planner then aligns reference points $\{(1, 4), (1, 5)\}$ without requiring any additional resolution actions. Finally, the planner attempts to align reference point $(1, 6)$. A simple alignment would stretch the interlink between cables 1 and 2; therefore **ResolveInterlinkConflict** is called to plan a resolution. The reference points re-

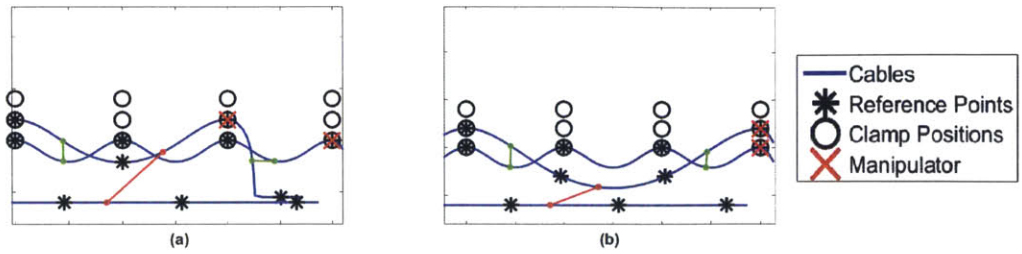


Figure 5-4: Infeasible Single step resolutions

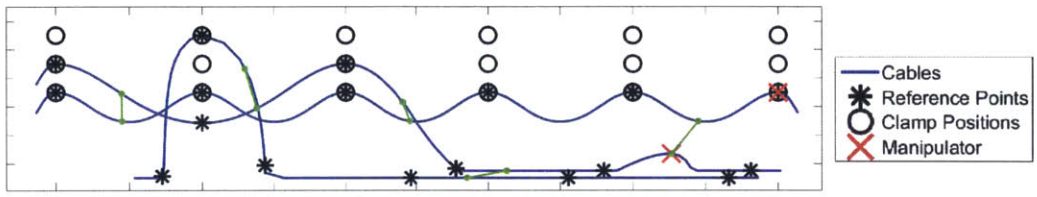


Figure 5-5: Interlink violation resolved through a cable repositioning

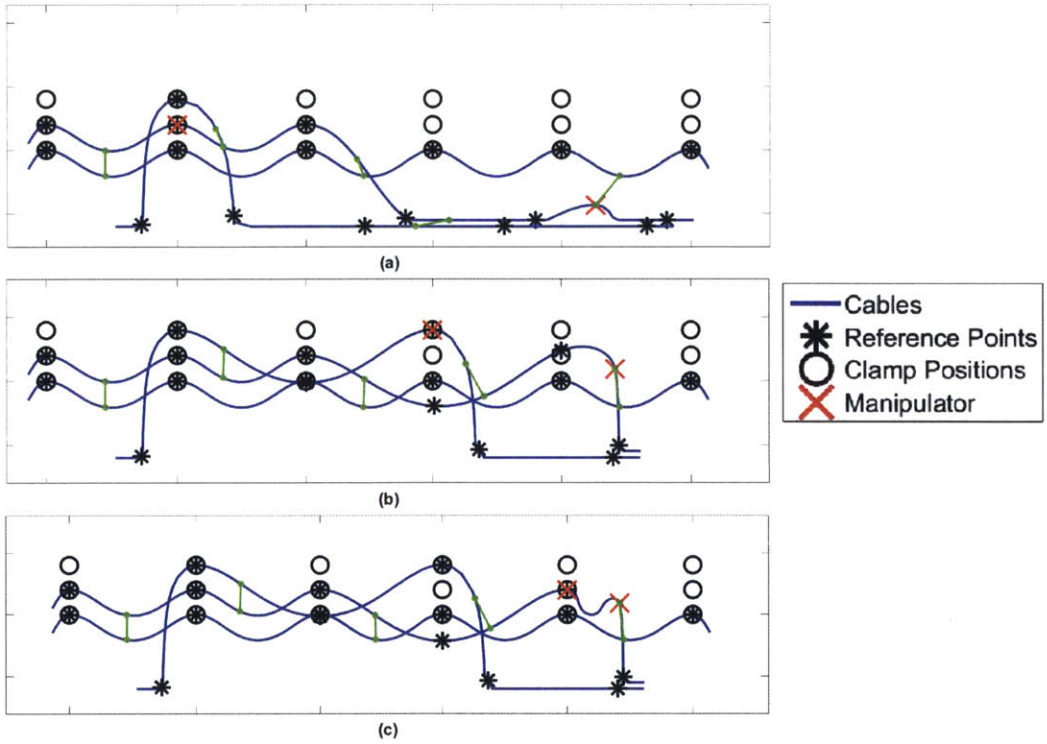


Figure 5-6: Alignment of additional reference points to ensure resolution

quired for single-step resolution are $\{(2,5),(2,6)\}$. However as depicted in figure5-4, single step resolution is infeasible with both the points, so **AttemptReposition-**

Resolution is called to plan a reposition-based resolution. Using **DetermineRefPointsToAlign**, it is determined that reference points $\{\mathbf{r}\} = \{(2,5),(3,4),(2,2)\}$ must be aligned to resolve interlink constraints while freeing all manipulators. The algorithm selected a grasp point on cable 2 for repositioning the cable. **ComputeGeometricResolutions** is then used to compute the manipulator position that ensures that the interlink is not stretched while aligning $(1,6)$, as shown in figure 5-3. Manipulator 1 is used to align the reference point, and manipulator 2 is used for the resolution action. **AlignReferencePointsList** is then called to align the list of reference points $\{\mathbf{r}\}$. Reference point $(2,5)$ cannot be aligned without stretching additional interlinks. If manipulator 1 is used to align $(3,4)$, manipulator 2 – which is already grasping cable 2 – can be used to reposition cable 2 in order to resolve one of the interlink conflicts. However, doing so would also stretch an interlink to the left of reference point $(2,3)$, which has already been clamped. Thus, manipulator 2 cannot influence the cable positions in that section, and aligning $(3,4)$ is not possible. However, the planner can align $(2,2)$ without requiring any additional actions, as shown in figure 5-6 (a). Once $(2,2)$ is aligned, the planner can then align $(3,4)$ while using manipulator 2 to reposition cable 2 as depicted in figure 5-6 (b). Finally, $(2,5)$ can be aligned, terminating the list $\{\mathbf{r}\}$. With the additional reference points aligned, manipulator 2 can now be freed with no taut interlinks. Alignment of all reference points in the original list is now completed, and the planner can exit successfully.

5.2 Application to Real World Problems

The motivating example for this paper was the automated installation of electrical cables into an aircraft fuselage. This process is typically performed manually in aircraft manufacturing: workers lay out the cables alongside the fuselage and align them with the appropriate clamping locations using temporary fasteners. They then secure the minor branches emerging from the main cables, and clamp the cables using permanent fasteners.

Initial cable placement is prone to errors during manual installation, and correct-

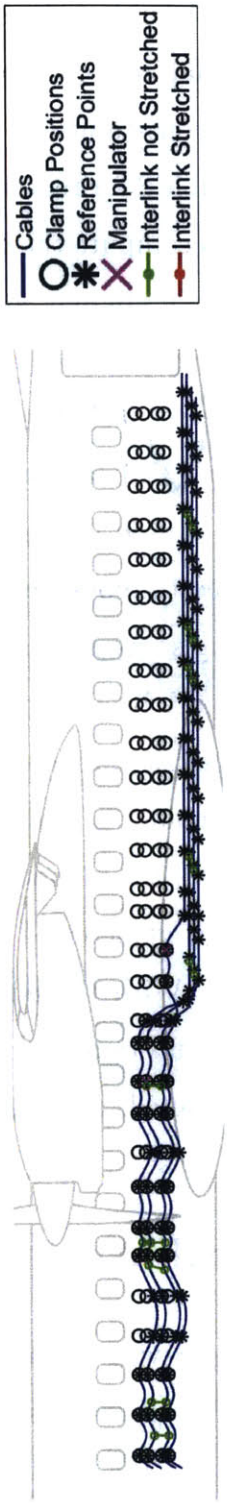


Figure 5-7: Cable installation in an aircraft fuselage

ing these errors is time-consuming. Automation of initial placement could reduce misalignment of cables while securing the minor branches. In this section, we present an instance modeling the installation process using the formulation provided. The cables, clamps and interlinks are modeled as per the actual dimensions in the fuselage.

The visualization of the system state during the problem is depicted in Figure 5-7. Each side of the fuselage has four cables with lengths of approximately 10 m. In the initial state, the cables $\mathbf{C}^i$ are assumed to be laid out straight along the fuselage floor. Each cable must be attached to the fuselage at 30 clamping locations per cable defined in the set. The clamp positions are known a priori and included in the set $\mathbf{K}^i$. If the distance between two clamp positions is x, the length of cable that must be between the clamps is $1.05x$. The positions of the reference points $\mathbf{R}^i$ are determined according to this slack factor. The gripping points $\mathbf{G}^i$ are assumed to be distributed uniformly along the length of each cable, with 625 gripping points per cable. The cable harness has 18 instances of interlinks between cables. Some interlinks repeat at set intervals throughout the fuselage, while some occur at only a single position. This placement results in varying interlink densities among different cable sections, as shown in Figure 5-7. Two manipulators are available to interact with the cables.

The planner presented in chapter 4, combined with the shape computation scheme presented in chapter 2, was implemented in MATLAB. The planner terminated successfully and generated a plan to complete the installation process in 224 temporally disjoint action sets, as defined in equation 3.11. The runtime for the planner was 1,102 seconds when executed on a machine with the Intel core i7-4702HQ CPU.

Out of 120 attempted reference point alignments, **ResolveInterlinkConflict** was called six times. Each time, single-step resolution successfully yielded a feasible resolution strategy, with no repositioning-based resolution attempts required. As expected, the planner never executed an action that would result in stretching of either a cable or an interlink. Prediction of the expected shape via the shape computation scheme ensured that potential interlink stretching was pre-empted and a resolution was planned before executing an action.

The proposed formulation captures aspects of modeling the cable installation pro-

cess that prior art has been unable to model. In addition, the proposed planner is capable of generating plans that successfully complete cable installation for real-world problems involving fewer manipulators than cables.

5.3 Analysis of Algorithm

In this section, we analyze the algorithm in terms of its time complexity and its completeness. The algorithm searches through a finite set of reference points for alignment. Additionally the resolution strategies involve manipulating a finite set of reference points and grip points. The computation of manipulator positions for repositioning-based resolution is repeated for a finite number of iterations, hence the algorithm is guaranteed to terminate. In 5.3.1 we show that the time complexity of the proposed planner is $\mathcal{O}(N_r^{2N_m+1} + N_r^{2N_m} N_g)$ where, N_r, N_g, N_m are the number of reference points, grip points and manipulators respectively.

The planning algorithm uses a fixed set of resolution strategies to generate interlink constraint resolution plans, namely single step resolution and repositioning-based resolution. However a larger set of resolution strategies exists which may generate solutions where the planner, with limited set of strategies, fails. In section 5.3.2 we discuss the nature of resolutions strategies required for a complete algorithm. In addition we provide examples where the proposed planner would fail to generate a solution.

5.3.1 Time Complexity of the Planner

AlignReferencePointList (algorithm 1) attempts to align a list of reference points. In the worst case, the only feasible order of alignment of reference points is reverse of the order provided at input. In that case the planner would require $N_r(N_r - 1)/2$ attempts to terminate, which is $\mathcal{O}(N_{ref}^2)$. In the worst case, every alignment requires interlink conflict resolution using algorithm 2. An interlink resolution would sequentially perform single step resolution, determine a list of additional reference points to align and then attempt reposition based resolution. The time complexity of

AlignReferencePointList when called with N_m manipulators ($T_{AR}(N_m)$) is given by

$$T_{AR}(N_m) = N_r^2[T_{SS}(N_m) + T_{DR} + T_{RR}(N_m)] \tag{5.1}$$

Where T_{SS} is the time complexity of single step resolution; T_{DR} is the time complexity of **DetermineRefPointsToAlign** (algorithm 4); T_{RR} is the time complexity of **AttemptRepositionResolution** (algorithm 3).

Single stage resolution attempts to align reference points on corresponding cables simultaneously. It searches through a set of reference points independently on all cables. The time complexity $T_{SS}(N_m)$ is given by

$$T_{SS}(N_m) = \mathcal{O}(N_m\ N_r) \tag{5.2}$$

In the worst case **DetermnineRefPointsToAlign** would add all the reference points to the align list. As each reference point is tried only once, they worst case time complexity would be

$$T_{DR} = \mathcal{O}(N_r) \tag{5.3}$$

AttemptRepositionResolution attempts reposition-based resolution with a combination of grip points on the corresponding cables. However it attempts repositioning the cable by successively selecting from the set of grip points independently for each cable. In addition to computing the manipulator positions for resolution, it also tries to align the list of reference points generated by **DetermnineRefPointsToAlign**. Hence the worst case time complexity would be

$$T_{RR}(N_m) = \mathcal{O}(N_m\ N_g) + T_{AR}(N_m - 1) \tag{5.4}$$

Substituting equations (5.2), (5.3), (5.4) in equation (5.1), we obtain a recursive

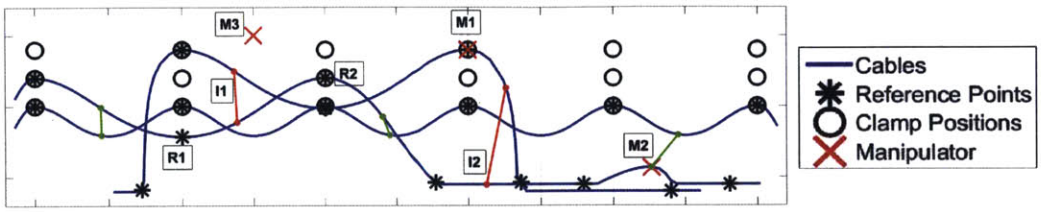


Figure 5-8: An instance where repositioning-based strategy fails.

functional expression

$$\begin{aligned} T_{AR}(N_m) =& N_{ref}^2[N_m\ N_r + N_r + N_{manip}\ N_g \\ &+ T_{AR}(N_m - 1] \end{aligned} \tag{5.5}$$

By expanding out the $T_{AR}(N_m - 1)$ term repeatedly, the expression can be written as a series sum given by

$$\begin{aligned} T_{AR}(N_m) =& \sum_{i=1}^{N_m}(N_r^2)^{i-1}[N_r^2\ N_m\ N_g + (-N_r^2\ N_g)(i-1)] \\ &+ \sum_{i=1}^{N_m}(N_r^2)^{i-1}[N_r^3\ N_m + (-N_r^3)(i-1)] \\ &+ \sum_{i=1}^{N_m}(N_r^2)^{i-1}(N_r^3) \end{aligned} \tag{5.6}$$

The first two terms of the equation (5.6) form an arithmatico-geometric series and the final term is a geometric series; the third term is a pure geomtric series. The summations of the terms are of $\mathcal{O}(N_r^{2N_m}\ N_g)$, $\mathcal{O}(N_r^{2N_m+1})$ and $\mathcal{O}(N_r^{2N_m+1})$ respectively. Thus we can show that the time complexity of the planner is

$$T_{AR}(N_m) = \mathcal{O}(N_R^{2N_m+1} + N_r^{2N_m}\ N_g) \tag{5.7}$$

5.3.2 Completeness of the planner

To ensure that the planner is complete, it must systematically and exhaustively search through all possible combinations of reference points for single step resolution, and grip points for repositioning-based resolution. A complete planner must also explore

through the possibility of assigning multiple manipulators to the same corresponding cable. In that case the number of possible selections would be $\mathcal{O}(\prod_{i=1}^{N_{cable}} \binom{N_r}{N_{m_i}})$ subject to $\sum_{i=1}^{N_{cable}} N_{m_i} = N_m$, where N_{m_i} is the number of manipulators assigned to a particular cable. The proposed algorithm does not carry out an exhaustive search required for completeness and is incomplete due to two reasons.

1. **Incomplete search in resolution strategies**:

   While performing single step resolution, and while selecting the grip points for repositioning-based resolution, the points are selected independently on each cables. The planner only looks for the point on a given cable that will resolve all interlinks that attach to that cable without stretching interlinks with any other cable. This approach requires $\mathcal{O}(N_m\ N_r)$ and $\mathcal{O}(N_m\ N_g)$ selections for single step resolution and repositioning-based resolution respectively. Unless, the interlinks are very densely distributed along the cable, the feasibility of a resolution strategy with the selected point depends more on the position of that point relative to the interlinks on that cable, than on selection of a point on another cable. The priority heuristic (4.1) ensures that points which are closest to the taut interlinks are selected first. Thus, the first few selections of reference points and grip points are likely to yield successful resolution plans. With this we trade-off completeness for the advantage of prioritizing selection of points on all corresponding cables most likely to yield a feasible resolution plan.

2. **Limited set of resolution strategies:**

   The next source of incompleteness is the limited set of strategies adopted to resolve interlink conflicts. In the adopted resolution strategies, at any given time a cable can only be manipulated by a single manipulator and vice versa. While completing the cable installation manually, the human workers often pick up and manipulate multiple cables at a time. The limited dexterity of robotic arms preclude manipulation of multiple cables with a single manipulator. Further, assigning only one manipulator to a cable for resolution prevents redundant manipulator assignments.

However, there are scenarios where this restriction would result in the algorithm not discovering a solution when one exists. Consider an instance as shown in figure 5-8. In this case, interlinks I1 and I2 are taut. The manipulator assigned to cable 2 is M2, thus the planner cannot assign any other manipulator to cable 2 for planning the resolution. However, cable 2 is clamped at reference point R2. Thus while M2 can be used to resolve the interlink constraint I2, it cannot influence interlink I1. Now manipulator M3 is free to grasp cable 2 between reference points R1 and R2 and try either single step resolution or repositioning-based resolution. However the constraint to assign one manipulator per corresponding cable will result in this possibility not being considered. A complete planner would need to reason over manipulator assignments to every segment of the cables between clamped reference points. As the cable installation progresses towards completion, the number of such independent cable segments increases. A complete planner must cope with the changing problem size as the task progresses towards completion.

Thus, the proposed algorithm is incomplete due to a non-exhaustive search for reference point and grip point selections for the resolution strategy and because of the limited set of resolution strategies.

Chapter 6

Conclusion

Final assembly tasks in aerospace and automotive manufacturing involving manipulation of deformable linear objects, like installation of cables and pipes are largely conducted manually. These taks are difficult to automate primarily due to unstructured task domains, lack of well defined subtask sequences and limits on dexterity of robotic manipulators. In this thesis we address the former two issues by tackling problem of task planning for installation of multiple interlinked DLOs to a support structure. The contribution of this thesis are threefold. We frame the prediction of shape of the DLO as a curve optimization problem and propose transformations linking geometrically similar shapes across the space of cable length and stiffness. We propose a novel mathematical definition for the manipulation planning problem that explicitly considers interlinks between the cables as constraints. Lastly, we propose a planning algorithm that, given the list of clamp positions to align, automatically generates the manipulation plan, and is guaranteed to terminate.

In this chapter we briefly summarize these contributions and propose directions for future research necessary to deploy an automated installation system in the real world.

6.1 DLO Shape Model

The problem of predicting the shape of a DLO given the boundary conditions has been studied in prior work as an optimal control problem. We extend the formulation to include the effect of gravity on the shape of the DLO. In the absence of gravity, the shape of the DLO normalized with respect to it's length is dependent only on the boundary conditions. However in presence of gravity, it depends on the relative values of the stiffness and linear mass density of the DLO, in addition to the boundary conditions. In section 2.2, we define the transformations that map geomtrically similar DLO shapes across the space of lengths and stiffness. In particular, we propose that the shape of a DLO with a length L and stiffness k is geometrically similar to the shape of a DLO with length L' and stiffness $k\frac{L'^3}{L^3}$, given identical linear mass densities and appropriately scaled boundary conditions. In section 2.3, we utilise these properties to develop a scheme for shape prediction, first for a two point boundary value problem, and later extend it to computation of the DLO shape with each clamp position and manipulator position defining a boundary condition. This scheme is then used in simulation to predict the shape of the cables in response to every primitive action.

6.2 Planning Problem Formulation

In chapter 3, we provide a complete mathematical formulation for the manipulation planning problem underlying the installation process. The state of the planning problem maintains a record of the shape of all cables in the problem, the boundary conditions defined by clamps or by manipulators and the state of each interlink constraint in the planning problem. We define a finite set of primitive manipulation actions which result in a change of state. Finally we define a structure of a manipulation plan that imposes strict simultaneity constraint on all primitive actions included in an *action set*, and temporal sequencing constraints on different *action sets*. The strict simultaneity of the primitive actions in the *action set* allow actions aimed at resolving the violations of interlink constraints.

6.3 Planning Algorithm

The state of the planning domain includes continuous variables describing the shape of the cables and the positions of the manipulators, in addition to the binary predicates describing clamp utilization and manipulator grasps. Thus, the described planning domain would be out of the scope of PDDL style domain definitions and the planners that act on such domains. The continuous extension, PDDL+, cannot deal with continuous variables described by differential constraints. Thus, we propose a planner, that produces manipulation plans in the format defined in section 3.4. It accepts a list of reference points to align as input, and attempts to align them with their respective clamps. If a simple alignment using a single manipulator is not possible, the planner uses two pre-defined resolutions strategies, single-step resolution and repositioning-based resolution to generate plans for resolving interlink constraint violations. The planner exits execution when either all reference points are aligned, or no additional reference point alignments are possible. In sections 5.1 and 5.2, we demonstrate the applications of the algorithm with a small scale example with three cables each having six clamping locations and two manipulators. This problem demonstrates both the resolution strategies employed by the planner. Next, we demonstrate automated task planning for installation of the electrical cables in an actual aircraft. The problem input was generated using specifications from an aicraft currently in production, with four cables, each having thirty clamping locations. Two manipulators were provided for completing the installation. The algorithm successfully planned the installation in 1,102 seconds when executed on an Intel core i7-4702HQ CPU. Thus we demonstrate that the proposed algorithm is capable of dealing with real-world sized problems. In section 5.3, we show that the algorithm is guaranteed to terminate with the worst-case time complexity $\mathcal{O}(N_R^{2N_m+1} + N_r^{2N_m}\ N_g)$. We also list the conditions necessary for a complete planner.

6.4 Future Work

The task planner proposed here has some limitations which should be addressed in future research. The shape computation module cannot handle scenarios where the DLO is in contact with environmental obstacles. The model also does not consider the effect of interlinks on the shape of the cables. In instances where the interlinks have a significantly less mass than main cable, these effects can be ignored. However it would not extend to the case of manipulating a two dimensional network of cables. The problem specification does not explicitly model the reach of the robotic manipulators, or the constraints restricting the pose of one of the manipulators due to the other manipulators. Finally, in the planning algorithm, the only pathological cases identified and resolved are violation of interlink constraints. However, other pathological instances like entangled cables and trapping of an object between a cable and an obstacle are not recognized. These cases may be avoided by convenient initial layout of the cables and clearing the workspace of unnecessary objects. Finally, the manipulation plan produced by the algorithm, generates only the task level actions with well defined geometric end points. The motion level component connecting the endpoints with paths has not been defined in this thesis.

Based on these limitations, we propose the following as viable future research directions

6.4.1 DLO Interactions With Environmental Obstacles

The shape computation is the most significant bottleneck for the planning algorithm. More efficient methods for shape computation can be developed using the method proposed by Bretl, [7] with modifications to incorporate the effect of gravity. Another direction in improving shape computation would be to use transcription methods developed by Rao et al. [43] and Patterson et al. [41] to allow for interaction of the cable with environmental obstacles. Generating good initial guesses to seed these methods remains a challenging task.

6.4.2 Hierarchical Task and Motion Planning

The proposed planner generates the manipulation plans at a task level while providing geometric constraints as end points for lower level motion planners. The planning algorithm does not consider any interaction with a motion level planner in generating its plans, and does not guarantee that a motion level refinement of the generated plan is possible. Approaches by Garrett et al. [15] and Srivastava et al. [48] have designed interaction layers between task and motion planners that guarantee a motion level refinement to a task plan. These approaches were developed in the pick and place domain and would need to be adopted for the manipulation planning domain. Further, the proposed planner does not include a notion of optimality. Future research may also focus on development of a planner that produces plans which are optimal with respect to time taken to completion or distance traveled by the robot during installation.

6.4.3 Hardware Challenges in Perception and Manipulation

Significant hardware challenges remain in physical manipulation of deformable linear object. Primary amongst them is the need to develop a reliable perception system to identify and localise a DLO in space. It would be of value to develop a system that may identify multiple DLOs from a point cloud of the scene. Additionally, due to the length of the cables in processes like final assembly, such a system must handle instances where the entire cable is not in the field of view of the sensor, and it must identify which part of the cable is being observed. Such a system must also identify the state of the system in terms of clamps used and points on the cable being manipulated.

Finally, the reliable grasp generation and development of robot hardware and appropriate temporary fasteners for attaching the cable to the clamp location would also be an interesting research direction

Appendix A

Derivation of Two-point Boundary Value Problem

As described in section 2.1, the curve optimization formulation at the heart of the shape prediction problem is given by

$$Localmin\ J = \int_0^1 k_1 \frac{u^2}{L^2} + k_2 \frac{\tau^2}{L^2} + LW_g y ds \tag{A.1}$$

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 $\mathbf{C}(1) = \mathbf{C_f}$ for fixed-fixed case

with boundary conditions $\mathbf{C}(0) = \mathbf{C_0}$ for the fixed-free case

Let the curve $\mathbf{C}(s)$ be a local minima of the optimization problem. The necessary conditions may be derived using the calculus of variations approach.

Let $\frac{d\mathbf{C}(s)}{ds} = \mathbf{f}(\mathbf{C}, \tau, u)$ represent the governing differential equations. The hamiltonian for the system is given by

$$
\begin{aligned}
H &= k_1 \frac{u^2}{L^2} + k_2 \frac{\tau^2}{L^2} + LW_g y + \boldsymbol{\lambda}^T \mathbf{f} \\
\boldsymbol{\lambda}(s) &= \begin{bmatrix} \lambda_1(s) & \lambda_2(s) & \lambda_3(s) & \lambda_4(s) & \lambda_5(s) & \lambda_6(s) \end{bmatrix}^T
\end{aligned}
\tag{A.2}
$$

$\boldsymbol{\lambda}$ represent the time varying co-states, with one corresponding to each state.

For a local minima, the selected control input must minimize the hamiltonian, hence $\tau(s)$ and $u(s)$ can be derived by setting $\frac{\partial H}{\partial \tau} = \frac{\partial H}{\partial u} = 0$. This gives the control inputs to be

$$
\begin{aligned}
u &= \frac{-L^3}{k_1} \sum_{i=4}^{6} \lambda_i \; g_{1_i}(\theta, \phi, \psi) \\
\tau &= \frac{-L^3}{k_2} \sum_{i=4}^{6} \lambda_i \; g_{2_i}(\theta, \phi, \psi)
\end{aligned}
\tag{A.3}
$$

The governing differential equations for the states and the co-states are given by

$$
\begin{bmatrix} \frac{d\mathbf{C}}{ds} \\ \frac{d\boldsymbol{\lambda}}{ds} \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\mathbf{C}, u, \tau) \\ \frac{\partial H}{\partial \mathbf{C}} \end{bmatrix}
\tag{A.4}
$$

Thus, the governing differential equations for the states and the costates are

$$
\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{dz}{ds} \\ \frac{d\phi}{ds} \\ \frac{d\theta}{ds} \\ \frac{d\psi}{ds} \\ \frac{d\lambda_1}{ds} \\ \frac{d\lambda_2}{ds} \\ \frac{d\lambda_3}{ds} \\ \frac{d\lambda_4}{ds} \\ \frac{d\lambda_5}{ds} \\ \frac{d\lambda_6}{ds} \end{bmatrix} = \begin{bmatrix} L\, cos(\theta) cos(\psi) \\ L\, cos(\theta) sin(\psi) \\ L\, sin(\theta) \\ \tau + u\, tan(\theta) sin(\phi) \\ u\, cos(\phi) \\ u\, sec(\theta) sin(\phi) \\ 0 \\ -L\, W_g \\ 0 \\ L\, f_1 + L^2\, f_1' \\ L\, f_2 + L^2\, f_2' \\ L\, f_3 + L^2\, f_3' \end{bmatrix}
\tag{A.5}
$$

where

$$
\begin{aligned}
f_k &= \sum_{i=1}^{3} \lambda_i \; f_{k_i}(\phi, \theta, \psi); k \in \{1, 2, 3\} \\
f'_k &= \sum_{i=4}^{6} \sum_{j=4}^{6} \lambda_i \; \lambda_j \; f_{,k_{ij}} \, (\phi, \theta, \psi, k_1, k_2); k \in \{1, 2, 3\}
\end{aligned}
\tag{A.6}
$$

The boundary conditions in the fixed-fixed case are already defined in the problem namely:

$$
\mathbf{C}(0) = \mathbf{C_0} \; and \; \mathbf{C}(1) = \mathbf{C_f} \tag{A.7}
$$

In case of fixed-free boundary conditions, the terminal co-states are defined by the terminal part of the objective functional J. As J does not depend on terminal state values, the terminal co-states are 0. Thus in the fixed-free case, the boundary conditions are given by

$$
\mathbf{C}(0) = \mathbf{C_0} \; and \; \boldsymbol{\lambda}(1) = \mathbf{0} \tag{A.8}
$$

Appendix B

Proof of Similarity Transform

Here we present a proof for the third property stated in section 2.2 which states that:

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 and the mass per unit length of the DLO is constant.

As derived in Appendix A, for a DLO with length L, stiffness k_1 and k_2 and some specified endpoint configurations, the boundary value problem that the shape curve is a solution to is given by

$$\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{dz}{ds} \\ \frac{d\phi}{ds} \\ \frac{d\theta}{ds} \\ \frac{d\psi}{ds} \\ \frac{d\lambda_1}{ds} \\ \frac{d\lambda_2}{ds} \\ \frac{d\lambda_3}{ds} \\ \frac{d\lambda_4}{ds} \\ \frac{d\lambda_5}{ds} \\ \frac{d\lambda_6}{ds} \end{bmatrix} = \begin{bmatrix} L\ cos(\theta)cos(\psi) \\ L\ cos(\theta)sin(\psi) \\ L\ sin(\theta) \\ \tau + u\ tan(\theta)sin(\phi) \\ u\ cos(\phi) \\ u\ sec(\theta)sin(\phi) \\ 0 \\ -L\ W_g \\ 0 \\ L\ f_1 + L^2\ f_1' \\ L\ f_2 + L^2\ f_2' \\ L\ f_3 + L^2\ f_3' \end{bmatrix} \tag{B.1}$$

where

$$
\begin{aligned}
u &= \frac{-L^2}{k_1} \sum_{i=4}^{6} \lambda_i \; g_{1_i}(\theta, \phi, \psi) \\
\tau &= \frac{-L^2}{k_2} \sum_{i=4}^{6} \lambda_i \; g_{2_i}(\theta, \phi, \psi) \\
f_k &= \sum_{i=1}^{3} \lambda_i \; f_{k_i}(\phi, \theta, \psi); k \in \{1, 2, 3\} \\
f'_k &= \sum_{i=4}^{6} \sum_{j=4}^{6} \lambda_i \; \lambda_j \; f'_{k_{ij}}(\phi, \theta, \psi, k_1, k_2); k \in \{1, 2, 3\}
\end{aligned}
\tag{B.2}
$$

subject to

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

It must be noted that the functions u and τ are linear in $\lambda_i; i \in \{4, 5, 6\}$. $f_k; k \in \{1, 2, 3\}$ are linear in $\lambda_i; i \in \{1, 2, 3\}$. $f'_k; k \in \{1, 2, 3\}$ are quadratic in $\lambda_i; i \in \{4, 5, 6\}$.

The property states that the solution to equation B.1 must be geometrically similar to the case where $L = 1$ and the stiffness parameters are $k_1 = \frac{k_1}{L^3}$ and $k_2 = \frac{k_2}{L^3}$, with the boundary conditions appropriately scaled. For the fixed-fixed case with the end point conditions $\mathbf{C}_0 = \begin{bmatrix} \mathbf{X}_0^T & \mathbf{\Phi}_0^T \end{bmatrix}^T$, $\mathbf{C}_f = \begin{bmatrix} \mathbf{X}_f^T & \mathbf{\Phi}_f^T \end{bmatrix}^T$. The scaled boundary conditions are

$$\mathbf{C}'(0) = \mathbf{C_0} \; and \; \mathbf{C}'(1) = \begin{bmatrix} \frac{(\mathbf{X}_f - \mathbf{X}_0)^T}{L} & \mathbf{\Phi}_f^T \end{bmatrix}^T \tag{B.3}$$

The objective functional for minimization would be

$$J = \int_0^1 \frac{k_1}{L^3} u^2 + \frac{k_2}{L^3} \tau^2 + W_g y ds \tag{B.4}$$

Using an approach identical to that described in appendix A, the boundary value problem corresponding to this curve optimisation problem is:

$$\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{dz}{ds} \\ \frac{d\phi}{ds} \\ \frac{d\theta}{ds} \\ \frac{d\psi}{ds} \\ \frac{d\lambda_1}{ds} \\ \frac{d\lambda_2}{ds} \\ \frac{d\lambda_3}{ds} \\ \frac{d\lambda_4}{ds} \\ \frac{d\lambda_5}{ds} \\ \frac{d\lambda_6}{ds} \end{bmatrix} = \begin{bmatrix} cos(\theta)cos(\psi) \\ cos(\theta)sin(\psi) \\ sin(\theta) \\ \tau + u\ tan(\theta)sin(\phi) \\ u\ cos(\phi) \\ u\ sec(\theta)sin(\phi) \\ 0 \\ -W_g \\ 0 \\ f_1 + L^3\ f_1' \\ f_2 + L^3\ f_2' \\ f_3 + L^3\ f_3' \end{bmatrix} \tag{B.5}$$

where

$$u = \frac{-L^3}{k_1} \sum_{i=4}^{6} \lambda_i\ g_{1_i}(\theta, \phi, \psi)$$

$$\tau = \frac{-L^3}{k_2} \sum_{i=4}^{6} \lambda_i\ g_{2_i}(\theta, \phi, \psi)$$

subject to

$$\mathbf{C}'(0) = \mathbf{C_0}\ and\ \mathbf{C}'(L) = \mathbf{C_f} \tag{B.6}$$

in the fixed-fixed case and

$$\mathbf{C}'(0) = \mathbf{C_0}\ and\ \boldsymbol{\lambda}(1) = \mathbf{0}$$

in the fixed-free case.

When the solution is scaled geometrically by a factor of L, it would satisfy the following boundary value problem

$$\begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \\ \frac{dz}{ds} \\ \frac{d\phi}{ds} \\ \frac{d\theta}{ds} \\ \frac{d\psi}{ds} \\ \frac{d\lambda_1}{ds} \\ \frac{d\lambda_2}{ds} \\ \frac{d\lambda_3}{ds} \\ \frac{d\lambda_4}{ds} \\ \frac{d\lambda_5}{ds} \\ \frac{d\lambda_6}{ds} \end{bmatrix} = \begin{bmatrix} L\ cos(\theta)cos(\psi) \\ L\ cos(\theta)sin(\psi) \\ L\ sin(\theta) \\ \tau + u\ tan(\theta)sin(\phi) \\ u\ cos(\phi) \\ u\ sec(\theta)sin(\phi) \\ 0 \\ -W_g \\ 0 \\ f_1 + L^3\ f_1' \\ f_2 + L^3\ f_2' \\ f_3 + L^3\ f_3' \end{bmatrix} \tag{B.7}$$

where

$$\begin{aligned} u &= \frac{-L^3}{k_1} \sum_{i=4}^{6} \lambda_i\ g_{1_i}(\theta, \phi, \psi) \\ \tau &= \frac{-L^3}{k_2} \sum_{i=4}^{6} \lambda_i\ g_{2_i}(\theta, \phi, \psi) \end{aligned} \tag{B.8}$$

subject to

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

in the fixed-fixed case and

$$\mathbf{C}'(0) = \mathbf{C_0}\ and\ \boldsymbol{\lambda}(1) = \mathbf{0}$$

in the fixed-free case.

Note that the functions $f_k; k \in \{1,2,3\}$ and $f_k'; k \in \{1,2,3\}$ are identical those in (B.2). Further, as defined in equation (B.2), that the functions u and τ are linear in $\lambda_i; i \in \{4,5,6\}$. $f_k; k \in \{1,2,3\}$ are linear in $\lambda_i; i \in \{1,2,3\}$. $f_k'; k \in \{1,2,3\}$ are quadratic in $\lambda_i; i \in \{4,5,6\}$. If we assume that the initial conditions

$$\mathbf{X_0} = \begin{bmatrix} x_0 & y_0 & z_0 & \phi_0 & \theta_0 & \psi_0 & \boldsymbol{\lambda_0}^T \end{bmatrix}^T$$

solves problem (B.1), then

$$\mathbf{X_0} = \begin{bmatrix} x_0 & y_0 & z_0 & \phi_0 & \theta_0 & \psi_0 & \frac{\boldsymbol{\lambda_0}^T}{L} \end{bmatrix}^T$$

solves problem (B.7) with an identical solution for $\begin{bmatrix} x(s) & y(s) & z(s) & \phi(s) & \theta(s) & \psi(s) \end{bmatrix}^T$.

This proves that the shape solutions for DLO with length L and stiffness parameters k_1 and k_2 are geometrically similar to the solutions for DLO with length L' and stiffness parameters $k_1 \frac{L'^3}{L^3}$ and $k_2 \frac{L'^3}{L^3}$ respectively, provided the boundary conditions are scaled appropriately.

Bibliography

[1] Jürgen Acker and Dominik Henrich. Manipulation of deformable linear objects: From geometric model towards program generation. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2005, pages 1541–1547, 2005.

[2] Jürgen Acker, Björn Kahl, and Dominik Henrich. Environment guided handling of deformable linear objects: From task demonstration to task execution. *VDI Berichte*, (1956):231, 2006.

[3] Dmitry Berenson. Manipulation of deformable objects without modeling and simulating deformation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4525–4532, 2013.

[4] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. *ACM Transactions on Graphics*, 27:1, 2008.

[5] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.

[6] Andy Borum and Timothy Bretl. The free configuration space of a kirchhoff elastic rod is path-connected. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2958–2964. IEEE, 2015.

[7] T. Bretl and Z. McCarthy. Quasi-static manipulation of a Kirchhoff elastic rod based on a geometric analysis of equilibrium configurations. *The International Journal of Robotics Research*, 33(1):48–68, 2013.

[8] Joel Brown, Jean-claude Latombe, and Kevin Montgomery. Real-time knot-tying simulation. *The Visual Computer*, 20:165–179, 2004.

[9] Yu-Kuang Chang and Alyn P Rockwood. A generalized de Casteljau approach to 3D free-form deformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, number d, pages 257–260, 1994.

[10] Elaine Cohen, Tom Lyche, and Richard Riesenfeld. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, 1980.

[11] Laurent D Cohen and Isaac Cohen. Finite Element Methods for Active Contour Models and Balloons for 2D and 3D Images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (November):1131–1147, 1993.

[12] V Duindam, J Xu, R Alterovitz, S Sastry, and K Goldberg. Three-dimensional Motion Planning Algorithms for Steerable Needles Using Inverse Kinematics. *International Journal of Robotics Research (IJRR)*, 29(June):789–800, 2010.

[13] Jieqing Feng, Lizhuang Ma, and Qunsheng Peng. A new free-form deformation through the control of parametric surfaces. *Computers and Graphics*, 20(4):531–539, 1996.

[14] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research (JAIR)*, 27:235–297, 2006.

[15] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. FFRob: An efficient heuristic for task and motion planning. *Algorithmic Foundations of Robotics XI*, 107:179–195, 2015.

[16] Russell Gayle, Paul Segars, M.C. Lin, and Dinesh Manocha. Path planning for deformable robots in complex environments. In *Proc. of Robotics: Science and Systems (RSS)*, pages 225–232, 2005.

[17] B. K. Hinds and J. McCartney. Interactive garment design. *The Visual Computer*, 6(2):53–61, 1990.

[18] S. Hirai and T. Wada. Indirect simultaneous positioning of deformable objects with multi-pinching fingers based on an uncertain model. *Robotica*, 18(1):3–11, 2000.

[19] Jörg Hoffmann. Ff: The fast-forward planning system. *AI magazine*, 22(3):57, 2001.

[20] Jörg Hoffmann. FF: The fast-forward planning system. *AI magazine*, 22:57–62, 2001.

[21] B. K. P. Horn. The Curve of Least Energy. *ACM Transactions on Mathematical Software (TOMS)*, 9(4):441–460, 1983.

[22] Shervin Javdani, Sameep Tandon, Jie Tang, James F O'Brien, and Pieter Abbeel. Modeling and perception of deformable one-dimensional objects. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1607–1614, 2011.

[23] Michael Kallay. Plane curves of minimal energy. *ACM Transactions on Mathematical Software (TOMS)*, 12(3):219–222, 1986.

[24] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[25] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[26] Jacek Kierzenka and Lawrence F Shampine. A bvp solver based on residual control and the maltab pse. *ACM Transactions on Mathematical Software (TOMS)*, 27(3):299–316, 2001.

[27] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2012.

[28] Andrew Ladd and Lydia E. Kavraki. Motion Planning for Knot Untangling. *Algorithmic Foundations of Robotics V*, (1):7–23, 2004.

[29] Frederick Li, Jianmin Zhao, Beta Lam, and Rynson Lau. An efficient method for simulating flexible connectors. *Journal of Multimedia*, 4(2):94–100, 2009.

[30] Achim Loock and E Schömer. A virtual environment for interactive assembly simulation: From rigid bodies to deformable cables. In *World Multiconference on Systemics, Cybernetics and Informatics*, pages 325—-332, 2001.

[31] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3684–3691. IEEE, 2014.

[32] Dennis Matthews and Timothy Bretl. Experiments in quasi-static manipulation of a planar elastic rod. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5420–5427, 2012.

[33] Mark Moll and Lydia E Kavraki. Path Planning for Deformable Linear Objects. *Transactions on Robotics, IEEE*, 22(4):625–636, 2006.

[34] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi. Knot planning from observation. In *2003 IEEE International Conference on Robotics and Automation*, volume 3, pages 3887–3892, 2003.

[35] Mustafa Mukadam, Andy Borum, and Timothy Bretl. Quasi-static manipulation of a planar elastic rod using multiple robotic grippers. In *IEEE International Conference on Intelligent Robots and Systems*, number Iros, pages 55–60, 2014.

[36] M Müller, Julie Dorsey, and L McMillan. Stable Real-time Deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 49 – 54, 2002.

[37] Dinesh K. Pai. STRANDS: Interactive simulation of thin solids using cosserat models. In *Computer Graphics Forum*, volume 21, pages 347–352, 2002.

[38] Sachin Patil and Ron Alterovitz. Interactive motion planning for steerable needles in 3D environments with obstacles. In *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*, pages 893–899, 2010.

[39] Sachin Patil, Jessica Burgner, Robert J. Webster, and Ron Alterovitz. Needle steering in 3-D Via rapid replanning. *Robotics, IEEE Transactions on*, 30(4):853–864, 2014.

[40] Sachin Patil, Jia Pan, Pieter Abbeel, and Ken Goldberg. Planning curvature and torsion constrained ribbons in 3D with application to intracavitary Brachytherapy. *Automation Science and Engineering, IEEE Transactions on*, 12:1332–1345, 2015.

[41] Michael A Patterson and Anil V Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[42] L. Piegl. Modifying the shape of rational B-splines. Part 1: curves. *Computer-Aided Design*, 21(8):509–518, 1989.

[43] Anil V Rao, David A Benson, Christopher Darby, Michael A Patterson, Camila Francolin, Ilyssa Sanders, and Geoffrey T Huntington. Algorithm 902: Gpops, a matlab software for solving multiple-phase optimal control problems using the gauss pseudospectral method. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):22, 2010.

[44] Olivier Roussel, Michel Ta, and Timothy Bretl. Motion Planning for a Deformable Linear Object. In *European Workshop on Deformable Object Manipulation*, pages 1–6, 2014.

[45] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Transactions on Robotics*, 23(6):1141–1150, 2007.

[46] Mitul Saha, Pekka Isto, and Jean Claude Latombe. Motion planning for robotic manipulation of deformable linear objects. In *Springer Tracts in Advanced Robotics*, pages 2478–2484, 2006.

[47] Ankit J. Shah and Julie A. Shah. Towards manipulation planning for interlinked deformable linear objects. In *Proc. of ICRA*. IEEE, 2016.

[48] Siddharth Srivastava, Lorenzo Riano, Stuart Russell, and Pieter Abbeel. Using Classical Planners for Tasks with Continuous Operators in Robotics. In *ICAPS Workshop on Planning and Robotics*, 2013.

[49] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 205–214. 1987.

[50] Jur van den Berg, Sachin Patil, Ron Alterovitz, Pieter Abbeel, and Ken Goldberg. LQG-based planning, sensing, and control of steerable needles. *Algorithmic Foundations of Robotics IX*, pages 373–389, 2010.

[51] T. Wada, S. Hirai, S. Kawamura, and N. Kamiji. Robust manipulation of deformable objects by a simple PID feedback. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, volume 1, pages 85–90, 2001.

[52] Hidefumi Wakamatsu and Shinichi Hirai. Static Modeling of Deformation Based on Differential Geometry. *International Journal of Robotics Research*, 23(3):293–311, 2004.

[53] Hidefumi Wakamatsu, Takumi Matsumura, and Shinchi Hirai. Dynamic Analysis of Rodlike Object Deformation towards Their Dynamic Manipulation. In *Intelligent Robots and Systems, Proceedings of the IEEE/RSJ International Conference on*, pages 196–201, 1997.

[54] Hidefumi Wakamatsu, Kousaku Takahashi, and Shinichi Hirai. Dynamic modeling of linear object deformation based on differential geometry coordinates. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2005, pages 1028–1033, 2005.

[55] Hidefumi Wakamatsu, Akira Tsumaya, Eiji Arai, and Shinichi Hirai. Planning of One-Handed Knot tying / Raveling Manipulation of Linear Objects. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE*, number April, pages 1719–1725, 2004.

[56] F Wang, E Burdet, R Vuillemin, and H Bleuler. Knot-tying with visual and force feedback for VR laparoscopic training. In *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, volume 6, pages 5778–81, 2005.

[57] Sun. Wen and Ron. Alterovitz. Motion Planning under Uncertainty for Medical Needle Steering Using Optimization in Belief Space. In *Proceedings of IROS 2014*, page pp. 1775...1781, 2014.