# Vision-based robotic manipulation of deformable linear objects

Jihong Zhu

# THESIS
# To obtain the doctoral degree

Delivered by **University of Montpellier**

Prepared within the doctoral school I2S
And from the research unit UM-CNRS LIRMM

Major in : **Micro-electronical systems and robotics**

Presented by : **Jihong Zhu**

# Vision-based Robotic Manipulation of Deformable Linear Objects

Defend on March 24, 2020 before the jury composed of

| | | | |
|---|---|---|---|
| François Chaumette | DR | INRIA | Reviewer |
| Maximo A. Roa | Researcher | DLR | Reviewer |
| Véronique Perdereau | PU | Sorbonne Université | Examiner |
| Andrea Cherubini | MCF-HDR | Université de Montpellier | Supervisor |
| Philippe Fraisse | PU | Université de Montpellier | Co-supervisor |
| Claire Dune | MCF | Université de Toulon | Invited |
| David Navarro-Alarcon | Assis. Prof. | PolyU Hong Kong | Invited |

# Abstract

In robotics, the area of deformable object manipulation receives far less attention than that of rigid object manipulation. However, many objects in real life are deformable. Research on deformable object manipulation is indispensable to equip robots with full manipulation dexterity. Deformable linear object (DLO) is one type of deformable objects that commonly presents in the industry and households, for instance, electrical cables for power transfer, USB cables for data transfer, or ropes for dragging and lifting equipment. In the context of H2020 VERSATILE, a project focusing on industrial automation using robots, we focus our research on DLO manipulation via visual feedback.

One characteristic of deformable object manipulation is that the object shape changes while being manipulated. Consequently, a research direction is to control the shape of the object during manipulation. We tackle the shape control problem by using vision. Initially, we parameterize the shape with Fourier series, estimate and update the interaction matrix online, and finally control the DLO shape.

In the subsequent research, instead of using human-defined features for parameterization, we let the robot automatically learn feature vectors from visual data. We propose a method that allows the robot to simultaneously generate a feature vector and the interaction matrix from the same data. Our approach requires minimum data for initialization. Learning and control can be done online in an adaptive manner. We can also apply the method to rigid object manipulation directly without modification.

Neither of the two frameworks requires camera calibration, and both are verified with simulation and real robotic experiments.

Another area of importance in deformable object manipulation is the utilization of external contacts. The object deformation is defined in a configuration space of infinite dimension. Nonetheless, the inputs from robots are limited. External contacts can and should be used for manipulating deformable objects. We take a practical scenario in the industry – cable routing with external contacts as the process to automate with our robot. We propose a planning algorithm that allows the robot to use contacts for shaping the cable and achieving the desired cable configuration. Real robotic experiments with different contact placement scenarios further validate the algorithms.

# Résumé

En robotique, la manipulation d'objets déformables reçoit moins d'attention que celle d'objets rigides. Pourtant, de nombreux objets dans la vie réelle sont déformables. La recherche sur la manipulation d'objets déformables est indispensable pour doter les robots d'une dextérité de manipulation totale. La difficulté majeure de ce problème est que déformation de l'objet a un espace de configurations de dimensions infinie, tandis que les entrées du robots sont limitées. Dans le cadre de VERSATILE, un projet H2020 axé sur l'automatisation industrielle à l'aide de robots, nous avons axé nos recherches sur la manipulation d'objets déformables linéaires (câbles) par retour visuel.

Une caractéristique de la manipulation des objets déformables est que la forme de l'objet change pendant la manipulation. Par conséquent, un problème important consiste à contrôler la forme de l'objet pendant la manipulation. Nous avons abordé le problème du contrôle de forme en exploitant le retour visuel.

Dans un premier temps, nous avons représenté la forme de l'objet avec une série de Fourier. Nous estimons et mettons à jour la matrice d'interaction en ligne, puis nous concevons le contrôleur pour contrôler la forme.

Ensuite, au lieu d'utiliser une caractéristique définie par l'humain pour le paramétrage, nous avons laissé le robot apprendre automatiquement les vecteurs de caractéristiques à partir des données visuelles. Nous proposons une méthode qui permet au robot de générer simultanément - et à partir des mêmes données - un vecteur de caractéristiques ainsi que la matrice d'interaction. Cette méthode nécessite un minimum de données pour l'initialisation. L'apprentissage et le contrôle peuvent être effectués en ligne de manière adaptative. Nous pouvons appliquer la même méthode à la manipulation d'objets rigides, directement et sans modification.

Ces deux travaux ne requièrent aucune calibration de la caméra et ont été validés avec des expérimentations de robotique réelle.

Un autre domaine d'importance dans la manipulation d'objets déformables est l'utilisation de contacts externes pour contrôler la forme de l'objet. Les contacts externes peuvent et doivent être utilisés pour la manipulation d'objets déformables. Nous considérons un scénario fréquent dans l'industrie - l'acheminement de câbles avec des contacts externes comme processus à automatiser avec notre robot. Nous proposons un algorithme de planification qui permet au robot d'utiliser des contacts pour deformer le câble et pour obtenir la configuration souhaitée. Des expériences robotiques réelles avec différents scénarios de placement de contacts permettent de valider nos algorithmes.

# Acknowledgement

I would like to thank Prof. Andrea Cherubini for his patience, support and guidance during the three years of my PhD. I also want to thank my two co-supervisors, Prof. Philippe Fraisse and Prof. Andre Crosnier for their advice and feedback on my research.

I would like express my gratitude to the members of my jury: Prof. François Chaumette, Prof. Véronique Perdereau, Dr. Maximo A. Roa, Dr. Claire Dune, and Dr. David Navarro-Alarcon for reading my thesis and participating in the defense. I am grateful for Dr. Benjamin Navarro, Dr. Robin Passama to their help in conducting the robot experiments. A special thank to Dr. Kai Pfeiffer whom prove-read my thesis and gave me valuable feedback.

During my PhD, I have visited ROMI lab at The Hong Kong Polytechnic University hosted by Dr. David Navarro-Alarcon for three and a half months. Despite the political instability of HongKong at that period, thanks to the kind arrangement and hospitality of David and members of the ROMI lab, I had a great and fruitful visit in Hong Kong.

I had the pleasure of meeting all the great colleagues at IDH: Dr. Stephane Caron, Dr. Sonny Tarbouriech, Dr. Osama Mazhar, Dr. Takahiro Ito, Kévin Chappellet, Anastasia Bolotnikova, Dr. Niels Dehio, Mohamed Djeha, Dr. Yukiko Osawa Akiyama, Julien Roux, Saeid Samadi and Dr. Yuquan Wang. I enjoyed our time together and thank you all for making my staying in Montpellier memorable (I hope I have included everyone). I learned much more beyond research from all of you.

A thank to Ms. Jing Dai for her encouragement, patience and support during my three year of PhD.

Last, but certainly not least, I want to thank my family whom unfortunately due to the virus outbreak are not able to attend my defense. I am grateful for their unconditional love and encouragement.

Without all these people this thesis would not have been possible.

*To my parents*

# Content

# List of Figures

# List of Tables

# List of Acronyms

**ACMI** Angular Contact Mobility Index

**AFFD** Animated Free-Form Deformation

**CAD** Computer-aided Design

**DFFD** Discontinuous Free Form Deformation

**DLO** Deformable Linear Object

**DVS** Direct Visual Servoing

**EFFD** Extended Free Form Deformation

**FEM** Finite Element Method

**FFD** Free Form Deformation

**HSV** Hue-Saturation-Value

**IBVS** Image-based Visual Servoing

**MSD** Mass-Spring-Damper

**PBVS** Position-based Visual Servoing

**PCA** Principal Component Analysis

**RGB** Red-Green-Blue

**ROI** Region of Interest

**SER** Shape Error Reduction

**SVD** Singular Value Decomposition

Chapter 1

# Introduction

Humans are dexterous in manipulation. From an evolution perspective, the transition to bipedal that free both hands for manipulation contributes significantly to human intelligence [Newman and Newman, 2015]. Similarly, for robots to develop intelligent behavior, one of the critical aspects is manipulation. Robotic manipulation, as a subfield of robotics, has been studied for over four decades now. Most of the works assume that the object is rigid – the object's shape stays unchanged during manipulation. However, humans also manipulate deformable objects, i.e., objects whose shape changes cannot be neglected. Figure 1.1 shows multiple examples of humans manipulating deformable objects. The ability to manipulate deformable objects is indispensable for robots to achieve full manipulation dexterity.

(a) Cable harness.

(b) Folding clothes.

(c) Making pastry.

(d) Picking fruits.

Figure 1.1: Examples of humans manipulating deformable objects.

Considering object deformation in robotic manipulation introduces new challenges in

modeling, sensing, and control. We elaborate on these challenges one by one.

The modeling of deformable objects is a research topic in computer graphics. A realistic model encompassing physical properties is often computationally heavy and not suitable for real-time simulation [Moore and Molloy, 2007]. Sensing of deformation is usually done with vision or force. The extraction of the shape changes from sensory data is non-trivial. Last but not least, deformation imposes new problems in manipulation control, as the shape of the object is not static anymore. One prominent problem with control is underactuation. The inputs from the robots are limited, but the object's deformation has infinite degree of freedom (DoF).

Needless to say, we cannot tackle all three challenges within this thesis. Rather, we focus on one specific topic that is the vision-based shape control of deformable linear objects (DLOs), such as cables, ropes, wires to name a few. We start with a framework for dual arm shape control, and then a generalized framework for both rigid and deformable object manipulation is formulated. We explore the use of environmental contacts in deformable object manipulation that enables robots to perform cable routing tasks.

## 1.1 Motivation

Robotic manipulation research has resulted in a huge number of methods and algorithms. Only a small fraction of these are dedicated to deformable objects. Automatic manipulation and shaping of deformable objects opens new doors for robotic applications in areas like: surgical operation [King et al., 2009], agriculture [Li et al., 2011], food making [Yamaguchi and Atkeson, 2016], household services [Bersch et al., 2011] and industrial automation [Qin et al., 2019].

This thesis is supported by the H2020 VERSATILE project[1] – a project which aims at developing advanced robotic manipulation capabilities in industrial environments. One of the most common deformable objects used in the industrial setting are DLOs that enable data and power transfer. We usually need to manage them in an organized way, which usually involves conforming them to designated shapes or configurations. Therefore, we work on shaping DLO via visual feedback. Taking a step forward, we also develop a generalized framework for both rigid and deformable object manipulation using vision.

## 1.2 Contribution

This thesis contributes to the state-of-the-art in deformable object (mainly DLOs) manipulation in terms of novel algorithms and applications for shape control, specifically:

- A cable shape control algorithm for a dual arm robot is presented in Chapter 4. The

[1] `https://versatile-project.eu/`.

chapter is based on the paper [Zhu et al., 2018] "Dual-arm Robotic Manipulation of Flexible Cables" [2].

- A unified object manipulation framework via visual feedback is presented in Chapter 5. The chapter is based on [Zhu et al., 2020b] "Vision-based Manipulation of Deformable and Rigid Objects Using Subspace Projections of 2D Contours" [3], under review.

- A robotic manipulation planning utilizes environmental contacts for shaping DLOs is presented in Chapter 6. The chapter is based on the paper [Zhu et al., 2020a] "Robotic Manipulation Planning for Shaping Deformable Linear Objects With Environmental Contacts" [4].

The videos of the experiments are all available on the Interactive Digital Humans (IDH) group's YouTube channel:

- Cable shape control: `http://y2u.be/DPl_d7lbL84`
- Unified object manipulation: `http://y2u.be/gYfO2ZxZ5KQ`
- Contact-based planning: `http://y2u.be/7CdNQ4R_wT0`

The work of this thesis also motivated a workshop at IROS 2020 on "Managing Deformation: A Step Towards Higher Robot Autonomy" [5] where I am the main organizer.

## 1.3 Outline

The rest of the thesis is organized as follows:

**Chapter 2:** provides related works on deformable object manipulation and vision-based control.

**Chapter 3:** introduces the basic concept of visual servoing, which we adopt in the rest of this thesis. In addition, it provides a classification of DLOs based on [Henrich et al., 1999]. In the last section, we develop a DLO model for validation of the proposed algorithms.

**Chapter 4:** discusses dual arm vision-based DLO manipulation using Fourier parameterization based on [Zhu et al., 2018].

**Chapter 5:** discusses a unified vision-based scheme for manipulation of both rigid and deformable object. The chapter is based on [Zhu et al., 2020b].

[2] `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8593780`
[3] `https://hal.archives-ouvertes.fr/hal-02558064/`
[4] `https://hal.archives-ouvertes.fr/hal-02303257/`
[5] `https://sites.google.com/view/madef-iros2020/home`

**Chapter 6:** introduces a planning framework that allows robots to perform the cable routing task with environmental contacts. The chapter is based on [Zhu et al., 2020a].

**Chapter 7:** concludes the thesis and gives perspectives on future research.

Chapter 2

# Related Work

In this chapter, we review works that relate to the topic of this thesis. In the first section, we explain the concept of deformation. Although in this thesis we adopt a model-free approach for manipulation, it is crucial to review the works on deformation modeling, thus give the rationale behind our choice. A survey on deformation modeling is presented in Section 2.2. We dedicate Section 2.3 to previous works on sensor-based deformable object manipulation. Section. 2.4 presents works on DLO manipulation. Since two manipulation frameworks (Chapter 4 and 5) in this thesis are based on visual servoing, in Section 2.5, we review research on visual servoing.

## 2.1 Deformation

Deformation in the context of the thesis denotes changes in the shape or size of an object due to an applied force[1]. Depending on the resulting shape of the object after removing the force, we can classify deformation as plastic, elastic, or elasto-plastic.

A plastic deformation means that the object remains in its deformed state (shape) after the force is relaxed. An elastic deformation, on the contrary, entails that the object returns to its original state (shape) after having removed the force. The elasto-plastic deformation is somewhere in between: the object does not return to its original shape but also does not stay in its deformed shape.

Stress ($\sigma$) and strain ($\epsilon$) can describe deformation. Stress refers to the force applied over an area and is measured in pressure units ($N \cdot m^{-2}$). Strain measures the change in length or angle of the object due to such force.

As shown in Fig. 2.1, the elasticity in deformation is measured by Young's modulus $\mathcal{E}$ [Askeland and Phule, 2003], which induces a linear relationship between stress and strain [Callister et al., 2007]:

$$\sigma = \mathcal{E}\epsilon \tag{2.1}$$

The Young's modulus measures stiffness of the object. A large $\mathcal{E}$ implies high stiffness and vice versa. It is a vital parameter to model elastic deformation, yet geometrical linearity is not appropriate for large deformations, because only small deformations can

[1] Deformation caused by temperature is not taken into account in this thesis.

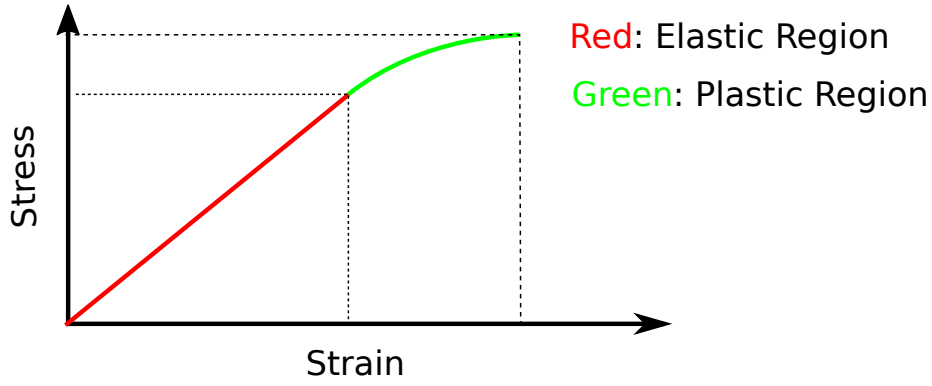



Figure 2.1: The difference between elastic and plastic deformation by relationship between stress and strain.

be modeled accurately [Nealen et al., 2006]. In the next section, we will discuss the means of modeling deformation.

## 2.2 Deformation Modeling

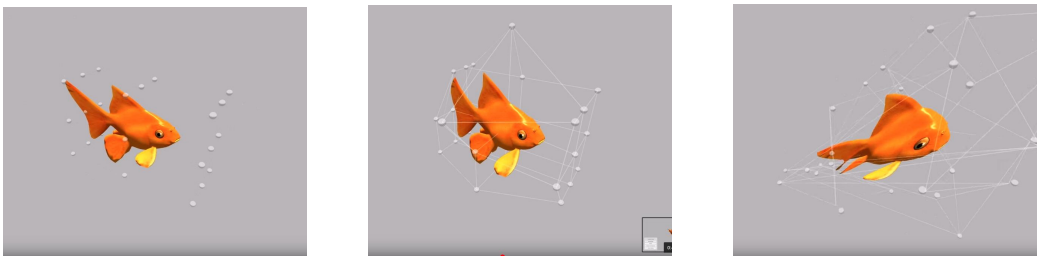
Deformation modeling is not a subject of robotics but rather of computer graphics. However, since in this thesis we are dealing with deformable objects, it is necessary to review models of deformation. In this section, we present the three most common models of deformation: i) pure geometric, ii) mass-spring-damper, iii) finite element method.

### 2.2.1 Pure Geometric Model

A pure geometric model, as its name suggests, does not take into account the physics that govern the deformation. The model consists of no knowledge of the mechanical property of the object. The model accuracy is compromised for fast computation.

The geometric model normally uses a set of control points to construct curves or surfaces. These control points are fitted with different spline functions, such as: Bezier curve, B-spline, and Fourier series among others. Examples of these methods can be found in [Bartels et al., 1987]. Instead of fitting control points, [Barr, 1987] introduced operators that can be applied to control points hierarchically to obtain deformation. A generalized method is later proposed by [Sederberg and Parry, 1986], which is known as the FFD method. The FFD encloses the object with a hull (usually a cube). The object deforms as the hull reshapes by the control points (see Fig. 2.2).

The FFD method received considerable attention in the research community. Researchers proposed several modified versions of FFD. Coquillart developed the Animated Free-Form Deformation (AFFD) [Coquillart and Jancene, 1991] and the Extended Free Form Deformation (EFFD) [Coquillart, 1990] that allowed more intuitive control over



(a) Original fish and enclosed cube with control points

(b) Deformed fish by pulling the control points

(c) Deformed fish by pulling the control points

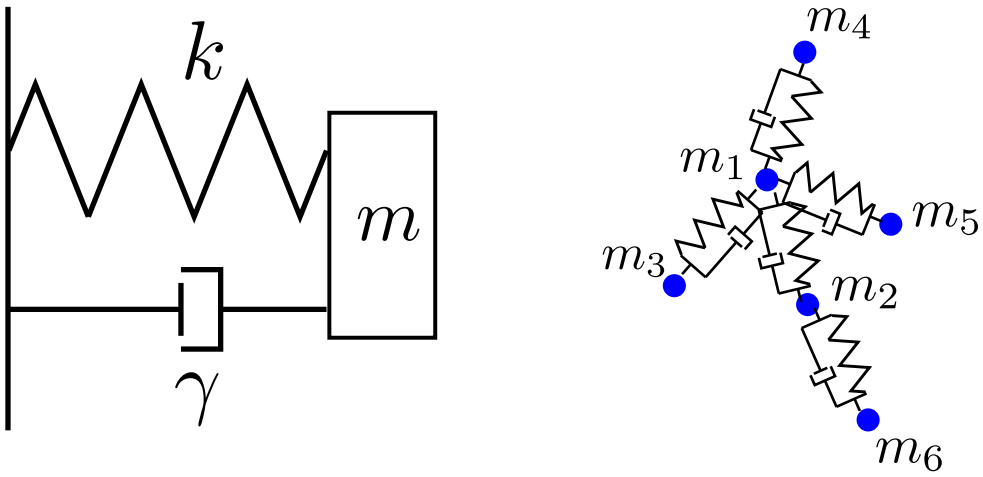
Figure 2.2: A demonstration of FFD method on an animated fish.

deformation. More recently, [Schein and Elber, 2004] developed the Discontinuous Free Form Deformation (DFFD) technique to better incorporate discontinuities in deformation.

The FFD and its extensions are typical geometric deformation models. They are used in Computer-aided Design (CAD) applications as a geometric sculpting tool. Since there is no information about the object's physical properties, the obtained model is not able to simulate the object realistically for real-time control purposes. For instance, the deformed fish in Fig. 2.2c in not realistic but can be obtained by manipulating the control points.

### 2.2.2 Mass-Spring-Damper (MSD) model

One simple dynamical model is the Mass-Spring-Damper (MSD) model. Instead of only considering the geometric points as in the previous section, the deformation model is represented by a collection of discretized point masses connected with springs and dampers in between (Fig. 2.3).




(a) Mass-spring-damper model

(b) Point masses connected by mass-spring-damper

Figure 2.3: Mass-spring-damper model explained.

The MSD model was initially used in facial modeling in computer graphics [Platt and Badler, 1981], [Waters, 1987]. Later, it was applied to the simulation of skin and muscle [Chadwick et al., 1989]. It can also be used to create animated locomotion for snakes, worms [Miller, 1988] and fishes [Tu and Terzopoulos, 1994]. A MSD-based cloth model is proposed in [Breen et al., 1994].

The MSD systems are intuitive, generally easy to implement, and computationally efficient, making real-time animations possible. The main drawback associated with using MSD systems is that the discrete model imposes significant approximations of the true physics that would occur in a continuous body [Moore and Molloy, 2007].

### 2.2.3 Finite Element Method (FEM)

Finite element methods divide the object into a discrete mesh of smaller components referred to as finite elements. The more elements the model has, the more computation is needed, and the more accurate the model will be. Compared to MSD methods which directly employ a discrete model, Finite Element Method (FEM) starts with continuous partial differential equations (PDEs) describing the deformation. These are then discretized over the single finite elements. The FEM result in a more accurate model, yet more computationally demanding.

In deformation modeling, the FEM is widely applied, to name a few applications: skin simulation [Benítez and Montáns, 2017], muscle [Chen, 1991], shape editing [Celniker and Gossard, 1991], and cloth modeling [Etzmuß et al., 2003]. FEM is used in areas where model accuracy is critical, such as surgical simulation [Berkley et al., 1999].

### 2.2.4 Summary

The above three methods for deformation modeling are ranked by model accuracy and computational load in Fig. 2.4.

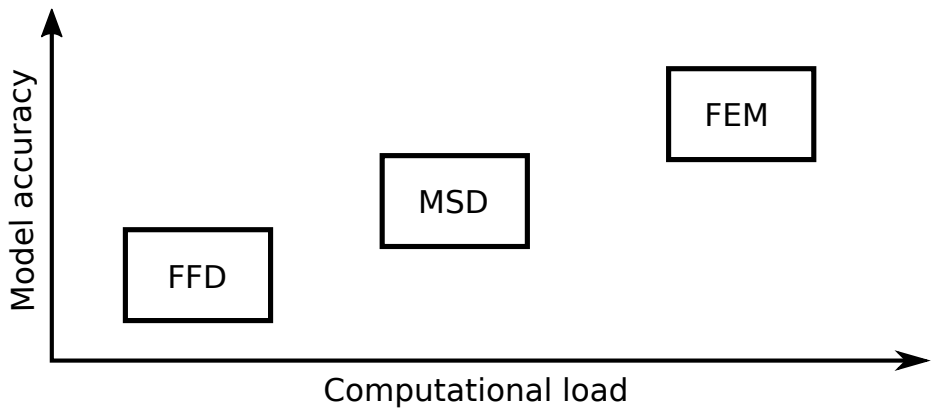



Figure 2.4: Comparison of three deformation modeling methods concerning computational load and model accuracy.

The FFD and its variations, since no physical property is involved, are not suitable for model-based robotic manipulation control of deformable objects. The two other methods are capable of simulating deformable objects accurately. However, the model needs to be identified before applying it to the real-time manipulation control. For different objects, a re-identification is required. If these methods are employed, we will need to construct offline different models before manipulating different objects. Therefore, instead of finding a global deformation model for robotic manipulation, we tend to favor model-free approaches. One prominent paper in this direction is [Berenson, 2013], where the author derived the interaction matrix based on the concept of diminishing rigidity. More recent approach, [Hu et al., 2019] learned the mapping between the robotic end-effector's movement and the object's deformation measurement with Gaussian process regression. Model-free approaches produce more general manipulation schemes with minimum offline modeling required. Besides, they are much less computationally demanding than MSD or FEM-based approaches.

Nevertheless, they also have their drawbacks, which we will later discuss in detail in Chapter 4 and 5.

## 2.3 Sensor-based Deformable Object Manipulation

Deformable object manipulation is an emerging area of research in the robotics community. We distinguish different manipulation approaches depending on the sensor feedback.

### 2.3.1 Tactile/force-based Manipulation

As mentioned in Section 2.1, deformation is a result of force applied to the object. Therefore, works have been done in manipulation based on tactile/force feedback.

Tactile information can be used to deduce deformation and thus feedback to control manipulation [Delgado et al., 2015]. Moreover, it can detect/correct slips and regulate grasping forces while manipulating deformable objects with a dynamic center of mass [Kaboli et al., 2016].

Due to critical tactile information produced during manipulation, research has been done in designing novel tactile sensors that can distinguish rigid and deformable objects [Drimus et al., 2014].

Force sensing could be useful in detecting vibration. For example, [Yue and Henrich, 2002] achieved fast manipulation under vibration with force/torque feedback. Some of the force-based methods make use of machine learning to derive manipulation strategies. A neural network was trained in [Howard and Bekey, 2000] to model the effect of force on deformable object manipulation. Recently, Lee et al. proposed a force-based manipulation skill learning approach for deformable object manipulation [Lee et al., 2015].

Since tactile and force information are often only locally available at contact points, these approaches are more focused on grasping the object or in-hand manipulation of the object. An object shape, instead, can represent the global deformation of that object. Shapes are usually observed via vision, therefore in the next section, we review works in vision-based manipulation.

### 2.3.2 Vision-based Manipulation

Another obvious indication of deformation is shapes. While force/tactile based approaches usually require a model to infer deformation as an outcome of the force applied, vision directly observes the resulting shape changes.

One of the initial works on the manipulation of deformable objects via visual feedback is presented in [Inoue, 1984] to solve a knotting problem. Smith et al. developed a relative elasticity model such that vision can be utilized without a physical model for the manipulation task [Smith et al., 1996]. Acker and Henrich applied vision to detect contact state changes [Acker and Henrich, 2003]. A hierarchical self-organizing neural network was developed along with vision to select proper grasping points on the deformable objects [Foresti and Pellegrino, 2004]. In recent research, Nair et al. combined learning and visual feedback to manipulate ropes [Nair et al., 2017]. Navarro-Alarcon et al. presented several papers focusing on vision-based deformable object manipulation. In his initial work, [Navarro-Alarcon et al., 2013a] employed the Broyden update rule for interaction matrix estimation, and a nonconservative Hamiltonian dynamical system to compute the state feedback laws. A passivity-based controller was proposed in [Navarro-Alarcon et al., 2013b] to deform object under inaccurate interaction matrix estimation. In this later work, [Navarro-Alarcon and Liu, 2014] considered 6 DoF motion of the robot manipulator for the manipulation task. However, the method used geometric features such as points, lines, and curvature to name a few, which could only express certain kinds of deformation. Taking a step further, in his latest work, [Navarro-Alarcon et al., 2016] proposed a generalized feature based on Fourier series for manipulation. [Laranjeira et al., 2017] proposed a vision-based management framework for tethers that link terrestrial mobile robots. Recently, [Chi and Berenson, 2019] introduced a robust vision-based sensing of deformable objects considering occlusion using Coherent Point Drift.

Other than receiving information locally, vision can provide global information on object deformation. Due to this aspect, vision often contains noisy data that needs pre-processing to yield a reasonable estimate of the deformation.

### 2.3.3 Multi-modal Manipulation

Since the problem of deformable object manipulation is complex, some researchers have also considered utilizing multiple sensor modalities for manipulation.

The author of [Hirai et al., 2001] presented one of the first multi-modal approaches for deformable object manipulation, where they used both force and vision information. In [Luo and Nelson, 2001], to observe changes in shapes, the authors applied an active contour method and a FEM model with force feedback, to predict deformation. Later [Huang et al., 2005] introduced a position/force hybrid control method that incorporates visual information with force control for flexible tool manipulation.

The multi-modal method is a promising research direction. Nonetheless, how to combine different sensor modalities to produce accurate state estimation of the deformable objects for feedback control is one of the challenges in this area of research.

## 2.4 Deformable Linear Object Manipulation

In this section, we review the type of research that is closely related to this thesis – DLO manipulation.

Compared with general deformable objects, DLOs are simpler to model. We distinguish two kinds of models: dynamical and topological. The former embeds physical properties of the objects and the latter concerns only the topology.

A dynamic model considering bend, twist, and extensional deformations is proposed in [Wakamatsu et al., 1995] using differential geometry. The shape of the DLO is solved by an optimization on the total energy under constraints. Later, the authour of [Wakamatsu and Hirai, 2004] used this model for grasping and manipulation of DLOs. The modeling can also be solved with FEM. The dynamic 2D deformation of an inextensible linear object was formulated using FEM in [Huang et al., 2008]. The method was claimed to be computationally faster than the differential geometry method mentioned previously. The author of [Yoshida et al., 2015] applied FEM for ring-shape objects for assembly tasks. Dynamical models can compute the shape of the DLO under constraints. These models are used when shape matters during manipulation.

Sometimes in DLO manipulation, topology rather than the exact shape is of interest, for instance, tying and untangling DLOs. Knot theory [Murasugi, 2007] can be used to develop a topological model for DLOs to solve the knotting problem. At an initial step in this direction, [Phillips et al., 2002] proposed a simple model for knot tying. In this model a rope in a loosely knotted configuration was pulled tight, and the knot was preserved, using an impulse model for collision handling. The rope was modeled as a spline of linear springs, with spheres placed on the control points to represent the rope volume. The spheres tend to bunch up or stretch apart during the simulation, due to the spring model, but collision handling did prevent the rope from passing through itself. In addition, the model did not operate in real time. The author of [Brown et al., 2004] took a step forward and formulated a knotting simulator capable of real-time simulation. These simulators contributed to the later works on robot motion planning to the tying

and untangle the DLO [Moll and Kavraki, 2006], [Ladd and Kavraki, 2004]. Researchers also explored learning in solving the knotting problem. The author of [Takamatsu et al., 2006] employed a learning from observation (LFO) paradigm for knot tying tasks. The robot learned a set of motions needed from human demonstration to complete the task. Knotting is of practical interest in medical and construction applications. A robotic knot tying framework in surgeries was presented in [Kang and Wen, 2002]. Quadrocopters performed aerial knot-tying which is an essential task in the aerial assembly of tensile structures [Augugliaro et al., 2015].

Several research works are dedicated to manipulation planning of DLOs: A collision-free path planner was developed in [Lamiraux and Kavraki, 2001] using a randomized algorithm. A planner in [Moll and Kavraki, 2006] computed a path in the shape space from one minimal energy curve to another while satisfying environmental constraints. Bretl and McCarthy showed that the shape space of an elastic rod is a six-dimensional smooth manifold [Bretl and McCarthy, 2014]. Later, the authors of [Borum and Bretl, 2015] took a step forward and proved the path-connectedness of this space.

## 2.5 Visual Servoing

Visual servoing or vision-based control is a crucial area of research in robotics. The use of vision enables robots to observe and actively react to the environment. The visual servoing scheme transforms visual information into a feature vector. The difference between the current feature vector value and the desired one produces an error term. Then the scheme tries to minimize this error by robot motion.

One of the first papers on visual servoing dates back to [Shirai and Inoue, 1973]. Since then, considerable effort has been made in the area. We refer interest readers to [Chaumette and Hutchinson, 2006a] and [Chaumette and Hutchinson, 2007a] for a comprehensive tutorial on this subject. There are two classes of the visual servoing scheme: Image-based Visual Servoing (IBVS) and Position-based Visual Servoing (PBVS). The former uses the image data directly to generate feature vectors while the latter uses image data for estimation of 3D position in space and then use it for the servoing task. Here we focus on IBVS, which is the method adopted in our research.

Some commonly selected geometric features for IBVS are points, lines and moments [Chaumette, 2004]. Kernel-based visual servoing [Kallem et al., 2007] combined feature tracking and control using spatial sampling kernels to generate features to design feedback controllers, thus eliminates the need for image processing to obtain features. Similar methods are referred to as Direct Visual Servoing (DVS). The DVS has received considerable attention. Several high impact works were published in the field. Collewet et al suggested using luminance of all pixels on the image for visual servoing [Collewet et al., 2008], [Collewet and Marchand, 2011]. In the subsequent research, photometric mo-

ments are considered as a new feature for visual servoing [Bakthavatchalam et al., 2013], [Bakthavatchalam et al., 2018]. In the latest research, [Marchand, 2019] proposed a DVS approach based on the Principal Component Analysis (PCA).

We intend to solve deformable object manipulation by IBVS. Therefore, methods and algorithms are developed in the framework of IBVS considering this context.

The authors of [Navarro-Alarcon and Liu, 2017] introduced the concept of shape servoing as a sub-field of visual servoing. The shape servoing research considers utilizing visual feedback for controlling the shape of the object. We extended the approach to consider a dual arm setup in [Zhu et al., 2018]. An expository paper on the topic is also available in [Navarro-Alarcon et al., 2019].

In this thesis, we extend and propose new algorithms and frameworks in shape servoing.

Chapter 3

# Preliminaries

In this chapter, we give a general introduction to key concepts of the thesis. Section 3.1 introduces a generalization of the visual servoing scheme [Chaumette and Hutchinson, 2006a], which we follow to develop our shape servoing algorithms. Section 3.2 addresses the concept of DLOs and their classification. We develop a model for verifying our shape servoing framework and present it in Section 3.2.2.

## 3.1 Visual Servoing

A unified description for all vision-based control is to minimize the error term $\boldsymbol{e}(t)$ in:

$$\boldsymbol{e}(t) = \boldsymbol{s}(\boldsymbol{m}(t), \boldsymbol{a}) - \boldsymbol{s}^*, \tag{3.1}$$

where $\boldsymbol{s}$ is the feature vector that consists of image measurements $\boldsymbol{m}$ and additional knowledge of the system $\boldsymbol{a}$ [Chaumette and Hutchinson, 2006a].

Visual servoing approaches differ in the selection of the feature vector $\boldsymbol{s}$. Once selected, we compute the relationship between robot spatial velocity $\dot{\boldsymbol{r}}$ and resulting time derivative of $\dot{\boldsymbol{s}}$:

$$\dot{\boldsymbol{s}} = \boldsymbol{L}\dot{\boldsymbol{r}}, \tag{3.2}$$

in which $\boldsymbol{L}$ is the interaction matrix (or feature Jacobian) relating the two. Given a time instant $\delta t$, we can write the discretized version of (3.2) as:

$$\delta\boldsymbol{s} = \boldsymbol{L}\delta\boldsymbol{r}, \tag{3.3}$$

where $\delta\boldsymbol{s} = \dot{\boldsymbol{s}}\delta t$ and $\delta\boldsymbol{r} = \dot{\boldsymbol{r}}\delta t$ correspond to the step changes in robot motion and feature vector respectively.

In general, it is not possible to obtain perfectly the interaction matrix. Thus its approximation $\hat{\boldsymbol{L}}$ is often used to formulate control:

$$\dot{\boldsymbol{r}} = -\lambda\hat{\boldsymbol{L}}^{\dagger}\dot{\boldsymbol{e}}, \tag{3.4}$$

where $^{\dagger}$ represents the Moore-Penrose pseudo-inverse. Similarly the discretized version of

(3.4) is:

$$\delta \boldsymbol{r} = -\lambda \hat{\boldsymbol{L}}^{\dagger} \boldsymbol{e}, \tag{3.5}$$

The shape servoing problem [Navarro-Alarcon and Liu, 2017] also falls into the formulation of (3.1). We will discuss it in detail in Chapter 4 and 5.

## 3.2 Deformable Linear Objects (DLOs)

Deformable Linear Objects commonly appear in household and industrial environment, for instance, ropes, elastic rods, beams, and cables to name a few. The term "Linear" refers to the shape of the object can be simplified as a curve/line as one dimension of the object is dominant in length.

### 3.2.1 Classification on Deformation Characteristics

Henrich et al. proposed a classification of DLOs based on their deformation characteristics [Henrich et al., 1999]:

Table 3.1: DLO classification with example objects reproduced from [Henrich et al., 1999].

| | **N** | **E-** | **E+** | **P-** | **P+** |
|---|---|---|---|---|---|
| Description | no deformation | low elastic deformation | high elastic deformation | low plastic deformation | high plastic deformation |
| Linear objects examples | short steal tubes | short spring steal | long spring steal | short iron wires | ropes |

The categorization between a + or - in Tab. 3.1 is dependent on whether gravity alone can make the object deform. If it is the case, then the object is in the + category; otherwise, it belongs to the - category.

The DLOs used in our experiment are in the **E+** category (marked with red color on the table).

### 3.2.2 Simulation Model of Deformable Linear Object

We adopt the model based on differential geometry [Wakamatsu and Hirai, 2004], where the shape of the DLO can be solved by a constrained optimization. We simplify it for our manipulation task by constraining the DLO to a 2D plane.

The deformation is expressed in terms of frame fields. Assume the total length of the linear object is $L$. $P(s)$ is a point at distance $s$ from the left end of the linear object. In Fig. 3.1, let $O - xy$ be the fixed world coordinate on the plane. $P - \zeta\xi$ is the local

coordinate attached to point $P(s)$. Angle $\phi(s)$ specifies the rotation of $P-\zeta\xi$ around the world frame $z$ axis.


(a) Natural state (b) Deformed state

Figure 3.1: Coordinate systems describe deformation.

The rotational matrix that transforms $O-xy$ into $P-\zeta\xi$ is:

$$A(\phi)=\begin{bmatrix}\cos\phi & -\sin\phi\\ \sin\phi & \cos\phi\end{bmatrix}$$

Let $\omega=\frac{d\phi}{ds}$ be an infinitesimal rotation at point $P(s)$. Then the following equation is satisfied:

$$\frac{dA}{ds}=A\begin{bmatrix}0 & -\omega\\ \omega & 0\end{bmatrix}$$

Since $A$ is an orthonormal matrix, we have then:

$$\begin{aligned}\begin{bmatrix}0 & -\omega\\ \omega & 0\end{bmatrix}&=A^T\frac{dA}{ds}\\ &=\begin{bmatrix}\cos\phi & \sin\phi\\ -\sin\phi & \cos\phi\end{bmatrix}\begin{bmatrix}-\sin\phi\frac{d\phi}{ds} & -\cos\phi\frac{d\phi}{ds}\\ -\cos\phi\frac{d\phi}{ds} & -\sin\phi\frac{d\phi}{ds}\end{bmatrix}\end{aligned}$$

The underlying principle in DLO modeling is that the potential energy of a deformable object reaches its minimum for the object's static shape [Wakamatsu et al., 1995]. In the 2D case, the total potential energy of the DLO is just the fluexual energy:

$$U=U_{\text{flex}},$$

which can be computed by integration:

$$U_{\text{flex}}=\frac{1}{2}\int_0^L R_f(\frac{d\phi}{ds})^2ds,$$

with $R_f$ a constant representing the flexural rigidity of the object.

Let us express $\phi(s)$ by a linear combination of basic functions $\boldsymbol{e}(s)$:

$$\phi(s)=\sum_{i=1}^{n}a_ie_i(s)=\boldsymbol{a}\cdot\boldsymbol{e}(s) \tag{3.6}$$

The basic function $\boldsymbol{e}(s)$ can be defined as:

$$\begin{aligned} e_1 &= 1, \\ e_2 &= s, \\ e_{2i+1} &= \sin\frac{2\pi i s}{L}, \\ e_{2i+2} &= \cos\frac{2\pi i s}{L}. \end{aligned}$$

During manipulation, four geometry constraints can be imposed on the DLO. They are the distance between two manipulation ends along the $x$ and $y$ axis: $l_x$ and $l_y$; plus two rotational angles at the two ends $\phi(0)$ and $\phi(L)$ as shown on Fig. 3.2.

The four constraints yields:

$$\begin{aligned} l_x &= \int_0^L \cos(\phi(s))ds, \\ l_y &= \int_0^L \sin(\phi(s))ds, \\ \phi(0) &= \theta_1, \\ \phi(L) &= \theta_2, \end{aligned} \tag{3.7}$$

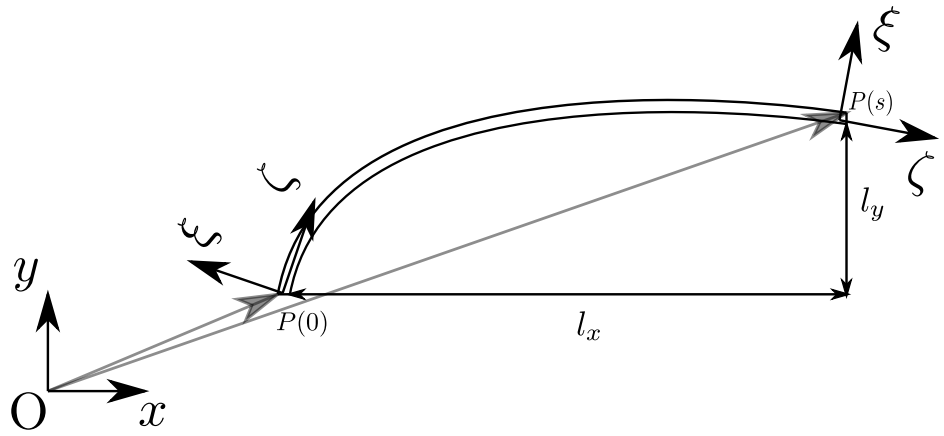



Figure 3.2: Geometrical constraints.

The final optimization problem is then:

$$\min_{\boldsymbol{a}} \frac{1}{2}\int_0^L R_f(\frac{d\phi(s)}{ds})^2 ds \tag{3.8}$$

subject to equality constraints in (3.7).

By solving the optimization problem, we obtain the shape of a DLO under specific manipulation constraints. We solve the optimization problem by CasADi [Andersson

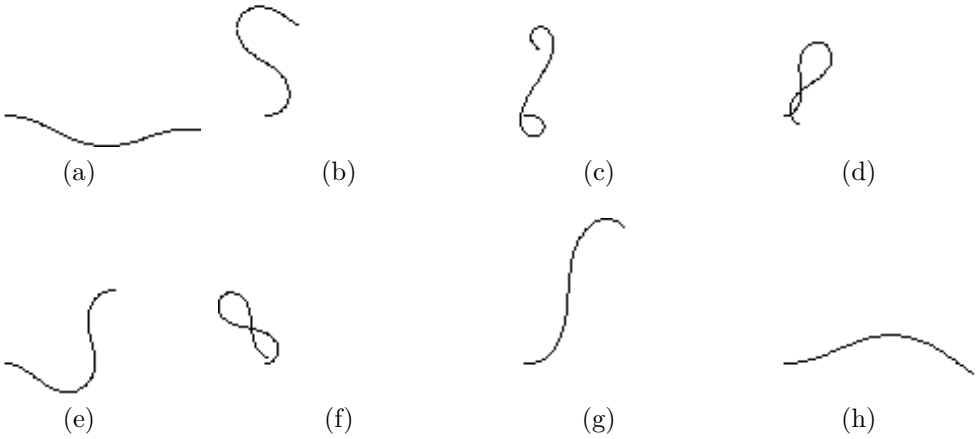



Figure 3.3: Example DLO shapes generated from the simulator

et al., 2019]. Figure 3.3 shows examples of shapes generated using our DLO simulator. Each cable is of unit length and represented by $K = 100$ points (tunable in the simulator). The simulator will be used later for testing the shape servoing frameworks [1]. Note that it is used only for verification of shape servoing algorithms, not for real-time simulation of any DLO.

[1]The 2D simulator of the DLO was made publicly available at `https://github.com/Jihong-Zhu/cableModelling2D`

Chapter 4

# Deformable Linear Object Manipulation with Fourier-based Features

As a starting point of the research, we focus on shaping DLO with a dual arm robot. We aim at a shaping task which humans can perform with both hands (see Fig. 4.1). In this chapter, we adopt the method in [Navarro-Alarcon and Liu, 2017] and propose a framework to deform a DLO to a target shape via visual feedback. For better readability, we replace the term DLO with cable in this chapter.

## 4.1 Introduction

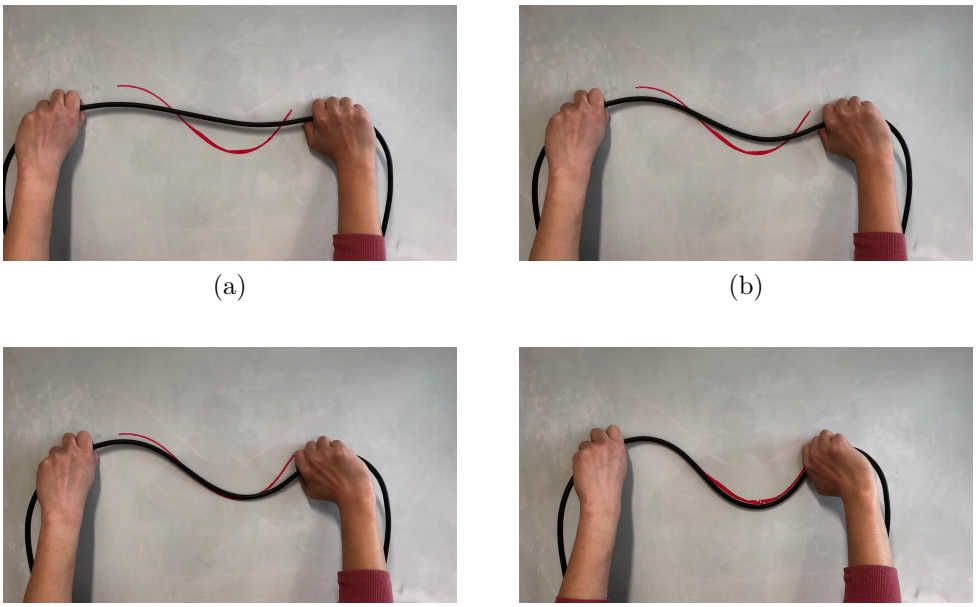
Given a (reachable) desired cable shape, a human can deform the cable into such shape relying on visual feedback without the need for re-grasping (see Fig. 4.1). This task is easy for a human to do without even knowing the internal dynamics of the cable. For robots, it remains a challenge.

### 4.1.1 Problem Statement

Let us consider a dual arm robot manipulating a cable on a 2D plane. The cable is a system with unknown dynamics that accepts inputs from the robot. Each robot end-effector applies three incremental inputs, respectively: two translations in the manipulation plane, $\delta x$ and $\delta y$, and one rotation $\delta\theta$ along the axis perpendicular to the manipulation plane (see Fig. 4.2). The total number of inputs from the robot is six, specifically:

$$\delta \boldsymbol{r} = [\delta x_1 \ \delta y_1 \ \delta\theta_1 \ \delta x_2 \ \delta y_2 \ \delta\theta_2]^T \in \mathbb{R}^6. \tag{4.1}$$

A static camera continuously observes the shape of the cable. The cable shape in the camera image is represented as $\boldsymbol{C} = [\boldsymbol{u}, \boldsymbol{v}]^T \in \mathbb{R}^{2\times K}$, where $\boldsymbol{u} \in \mathbb{R}^K$ and $\boldsymbol{v} \in \mathbb{R}^K$ are image coordinates of pixels sampled along the cable. We represent the desired cable shape by $\boldsymbol{C}^*$.



(a) (b) (c) (d)

Figure 4.1: Cable manipulation by humans(red color marks the desired shape).

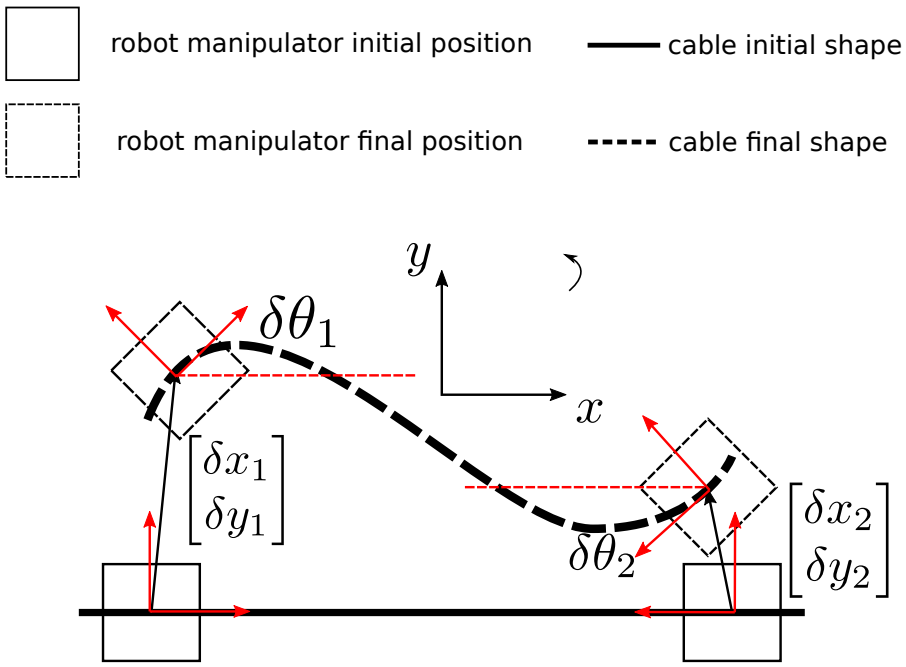



Figure 4.2: Control inputs for dual arm cable manipulation.

The problem we address is to use the control inputs $\delta \boldsymbol{r}$ to drive the cable from its initial shape $\boldsymbol{C}_0$ to the desired shape $\boldsymbol{C}^*$ using visual feedback.

## 4.2 Our Contributions

We propose a shape servoing algorithm to complete the task depicted in Fig. 4.1 for a dual arm robot. A receding window approach ensures that the most recent data is used for computing the interaction matrix relating the robot inputs and the shape changes. Then this interaction matrix is used to compute the robot inputs. The algorithm is validated by simulations and on a real robot setup.

## 4.3 Methods

To tackle the problem stated above, we first transform the cable shape into a feature vector. After parameterizing the shape, we model the relationship between the robot motion and the changes in the shape parameters locally by an interaction matrix. The final step is to derive the control strategy to deform the cable into the desired shape based on the interaction matrix. In this section, we describe each sub-task sequentially. The overall shape servoing scheme is explained in Fig. 4.3.

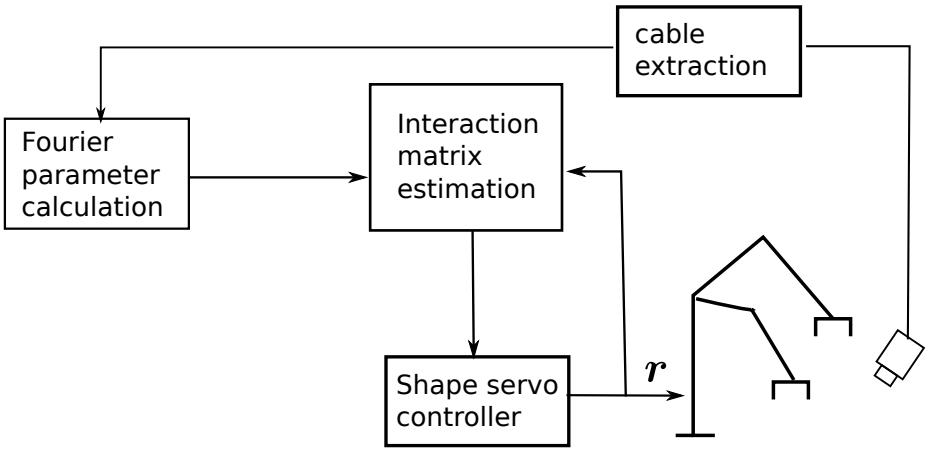



Figure 4.3: The cable manipulation scheme with a dual arm robot.

### 4.3.1 Fourier-based Features

The $i^{th}$ sample point $\boldsymbol{c}(i) = [u(i), v(i)]^T,\ i = 1, 2, \ldots, K$ in $\boldsymbol{C}$ can be approximated using a $N^{th}$ ordered Fourier series:

$$\begin{aligned} u(i) &= \sum_{j=1}^{N} [a_j\ b_j] \begin{bmatrix} cos(j\rho_i) \\ sin(j\rho_i) \end{bmatrix} + e, \\ v(i) &= \sum_{j=1}^{N} [c_j\ d_j] \begin{bmatrix} cos(j\rho_i) \\ sin(j\rho_i) \end{bmatrix} + f, \end{aligned} \tag{4.2}$$

with

$$\rho_i = (i-1)\frac{\pi}{K}.$$

The feature vector $\boldsymbol{s}$ in (4.2) is:

$$\boldsymbol{s} = [a_1\ b_1\ \ldots\ a_N\ b_N\ e\ c_1\ d_1\ \ldots\ c_N\ d_N\ f]^T \in \mathbb{R}^{4N+2}. \tag{4.3}$$

This will later be used both in deformation model estimation and control. Below we show how to solve for $\boldsymbol{s}$ given image data.

We can rewrite (4.2) as:

$$\boldsymbol{c}(i) = \begin{bmatrix} u(i) \\ v(i) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}^T(i) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{f}^T(i) \end{bmatrix} \boldsymbol{s}. \tag{4.4}$$

In (4.4), $\boldsymbol{f}(i)$ are the harmonics terms defined as:

$$\boldsymbol{f}(i) = [\cos\rho_i\ \sin\rho_i\ \ldots\ \cos(N\rho_i)\ \sin(N\rho_i)\ 1] \in \mathbb{R}^{2N+1}. \tag{4.5}$$

Using all $K$ samples in $\boldsymbol{C}$, we have:

$$\boldsymbol{C}' = \boldsymbol{G}\boldsymbol{s}, \tag{4.6}$$

with:

$$\begin{aligned} \boldsymbol{C}' &= [\boldsymbol{c}(1)^T\ \boldsymbol{c}(2)^T\ \ldots\ \boldsymbol{c}(K)^T]^T \in \mathbb{R}^{2K}, \\ \boldsymbol{G} &= \begin{bmatrix} \boldsymbol{f}(1) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{f}(1) \\ \vdots & \vdots \\ \boldsymbol{f}(K) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{f}(K) \end{bmatrix} \in \mathbb{R}^{2K\times(4N+2)}. \end{aligned}$$

In (4.6), $\boldsymbol{C}$ and $\boldsymbol{G}$ are known quantities. Therefore the feature vector $\boldsymbol{s}$ of shape $\boldsymbol{C}$ can

be solved by linear least squares:

$$\boldsymbol{s} = \boldsymbol{G}^{\dagger}\boldsymbol{C},$$

where $^{\dagger}$ represents the Moore-Penrose pseudo inverse: $\boldsymbol{G}^{\dagger} = (\boldsymbol{G}^T\boldsymbol{G})^{-1}\boldsymbol{G}^T$.

To ensure the pseudo inverse of $\boldsymbol{G}$ exists we need that:

$$\text{rank}(\boldsymbol{G}) = 4N + 2 \tag{4.7}$$

is fulfilled. A necessary condition for (4.7) is:

$$2K \geq 4N + 2.$$

So we need to collect at least $2N + 1$ samples to compute the feature vector.

### 4.3.2 Deformation Model Estimation

Feature vector $\boldsymbol{s}$ describes the cable shape. Locally, a small movement of the robot produces a tiny change in the cable shape, hence on the feature vector. From this observation, at a given operating point, we can linearize the deformation model as:

$$\delta\boldsymbol{s} = \boldsymbol{L}\delta\boldsymbol{r}. \tag{4.8}$$

Vector $\delta\boldsymbol{r} \in \mathbb{R}^6$ corresponds to robot motion, and $\delta\boldsymbol{s} \in \mathbb{R}^{4N+2}$ is the corresponding change in the feature vector; $\boldsymbol{L} \in \mathbb{R}^{(4N+2)\times 6}$ is the local deformation matrix.

For the $i^{th}$ element of $\boldsymbol{s}$, we can write:

$$\delta s_i = \delta\boldsymbol{r}^T\boldsymbol{l}_i^T,$$

where $\boldsymbol{l}_i$ is the $i^{th}$ row of $\boldsymbol{L}$. Its transpose $\boldsymbol{l}_i^T \in \mathbb{R}^6$.

To estimate $\boldsymbol{l}_i^T$, at the current instance $t_m$, we collect $M$ prior data of $\delta s_i$ and $\delta\boldsymbol{r}^T$ while the robot is moving:

$$\boldsymbol{\sigma}_i = \begin{bmatrix} \delta s_i(t_1) \\ \delta s_i(t_2) \\ \vdots \\ \delta s_i(t_M) \end{bmatrix} \in \mathbb{R}^M, \boldsymbol{R} = \begin{bmatrix} \delta\boldsymbol{r}^T(t_1) \\ \delta\boldsymbol{r}^T(t_2) \\ \vdots \\ \delta\boldsymbol{r}^T(t_M) \end{bmatrix} \in \mathbb{R}^{M\times 6}.$$

In the above equations, each $s_i(t_j)$ is the corresponding feature vector after robot motion $\delta\boldsymbol{r}^T(t_j)$. Using $\boldsymbol{R}$ and $\boldsymbol{\sigma}_i$ we have:

$$\boldsymbol{\sigma}_i = \boldsymbol{R}\boldsymbol{l}_i^T, \tag{4.9}$$

and we can estimate $\boldsymbol{l}_i$ as

$$\hat{\boldsymbol{l}}_i^T = (\boldsymbol{R}^T\boldsymbol{R})^{-1}\boldsymbol{R}^T\boldsymbol{\sigma}_i. \tag{4.10}$$

Therefore, we can estimate every row of $\boldsymbol{L}$. To successfully estimate $\boldsymbol{L}$, the following conditions must be satisfied:

- $\boldsymbol{R}$ must have full row rank.

If the condition is not satisfied, $\boldsymbol{R}^T\boldsymbol{R}$ in (4.10) will be singular. In practice, this condition infers that within the time window $\{t_1 - t_N\}$, it is necessary that the robot moves in all directions and rotates at least once. The condition makes perfect sense as if one or more component(s) of $\delta\boldsymbol{r}$ is/are not active during the whole time period, and no relationship can be derived between that/those component(s) and the feature vector.

To ensure that $\boldsymbol{R}$ has a full row rank, we can check the rank of $\boldsymbol{R}$ before the estimation of $\boldsymbol{L}$. If $\boldsymbol{R}$ does not fulfill the rank condition, we can:

- extend the time period until the rank condition is fulfilled or,
- apply Tikhonov regularization:

$$\hat{\boldsymbol{l}}_i^T = (\boldsymbol{R}^T\boldsymbol{R} + \lambda\boldsymbol{I})^{-1}\boldsymbol{R}^T\boldsymbol{\sigma}_i. \tag{4.11}$$

Last but not least, the time window $\{t_1 - t_m\}$ should be as short as possible while satisfying both conditions to ensure that the local interaction matrix is valid.

### 4.3.3 Shape Servo Controller

Using the camera, we observe the current parametric shape of cable $\boldsymbol{C}$. Given the target shape $\boldsymbol{C}^*$, both can be transformed to feature vectors $\boldsymbol{s}$ and $\boldsymbol{s}^*$. The differences between the current shape $\boldsymbol{C}$ and the desired shape $\boldsymbol{C}^*$ can be characterized by the difference between the feature vector of the two shapes:

$$\delta\boldsymbol{e} = \boldsymbol{s}^* - \boldsymbol{s}.$$

Using the estimated deformation model:

$$\delta\boldsymbol{s} = \hat{\boldsymbol{L}}\delta\boldsymbol{r},$$

and consider $\delta\boldsymbol{s} = \delta\boldsymbol{e}$

$$\delta\boldsymbol{r} = \hat{\boldsymbol{L}}^{\dagger}\delta\boldsymbol{e}, \tag{4.12}$$

The interaction matrix is a result of linearization in the vicinity of the current shape. Therefore, a relatively small motion is preferred. To prevent excessive motion produced

by (4.12), we calculate the input as:

$$\delta \boldsymbol{r} = \begin{cases} \delta \boldsymbol{r}, & \text{if } ||\delta \boldsymbol{r}||_2 < \boldsymbol{r}_{\max} \\ \boldsymbol{r}_{\max} \frac{\delta \boldsymbol{r}}{||\delta \boldsymbol{r}||_2}, & \text{otherwise} \end{cases}, \tag{4.13}$$

where $\boldsymbol{r}_{\max}$ is a positive scalar which serves as a saturation.

### 4.3.4 Receding Window Adaptation

Since equation (4.8) calculates a local interaction matrix, throughout manipulation, $\boldsymbol{L}$ needs to be updated.

At instance $t_N$, we estimate the local interaction matrix $\boldsymbol{L}_N$. Using the interaction matrix, we can derive the one-step command $\delta \boldsymbol{r}_N$ by (4.13). At $t_{N+1}$ instance, we execute the motion $\delta \boldsymbol{r}_N$. The cable is driven to a new shape with a new feature vector $\boldsymbol{s}(t_{N+1})$. Therefore, we have a new set of data:

$$\begin{cases} \delta \boldsymbol{s}(t_{N+1}) & = \boldsymbol{s}(t_{N+1}) - \boldsymbol{s}(t_N), \\ \delta \boldsymbol{r}(t_N) \end{cases} \tag{4.14}$$

We add in the newly obtained data (4.14), and remove the first data $\delta \boldsymbol{s}(t_1)$ and $\delta \boldsymbol{r}(t_1)$. The data window is shifted from $\{t_1 - t_N\}$ to $\{t_2 - t_{N+1}\}$. Using the new data window we can follow the same step and update $\boldsymbol{L}_N$ to be $\boldsymbol{L}_{N+1}$.

The receding window approach makes sure that, at any instance, we are using the newest data for estimating the interaction matrix.

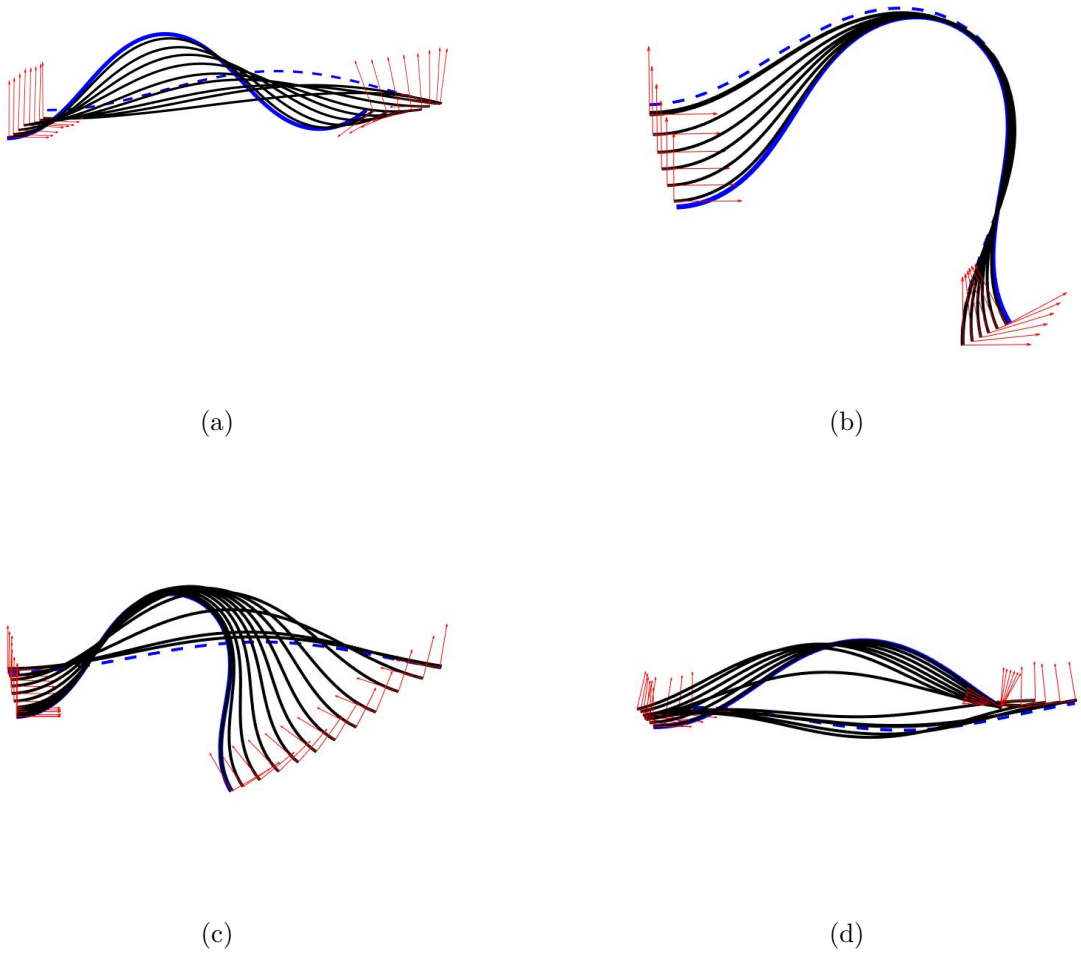
## 4.4 Simulation

Using the DLO model developed in Chapter 3.2.2, we simulate the proposed framework. The robot is holding both ends of the cable, thus imposes the constraints described in (3.7) for the optimization problem:

$$\begin{aligned} l_x &= \int_0^L \cos(\phi(s)) ds, \\ l_y &= \int_0^L \sin(\phi(s)) ds, \\ \phi(0) &= \theta_1, \\ \phi(L) &= \theta_2, \end{aligned} \tag{4.15}$$

The shape is solved with optimization. Our framework uses the computed shape from the simulator (represented by a set of points) as the cable shape $\boldsymbol{C}$, and the position and orientation of both ends of the cable as robot inputs. The new inputs are computed by

the framework, and then we put them as constraints for the optimization to solve for a new shape.



(a) (b)

(c) (d)

Figure 4.4: Manipulation simulation on the cable model. The initial and target shape are marked with dashed and solid blue, all the intermediate shapes are painted in black. The red coordinates at two endpoints specify the rotation of two end-effectors.

In Fig. 4.4, we present four examples of the evolution of the cable shapes under the proposed framework. The initial and target shape are marked with dashed and solid blue. The intermediate shapes are painted in black. The red coordinates at two endpoints specifying the rotation of two end-effectors.

The simulation shows that the controlled motion can successfully manipulate the cable to the target shape. However, in the simulation, we consider every component in the scheme is perfect and well synchronized, which is not always the case when it comes to robot experiments. Therefore, in the next section, we show results from real robot experiments.

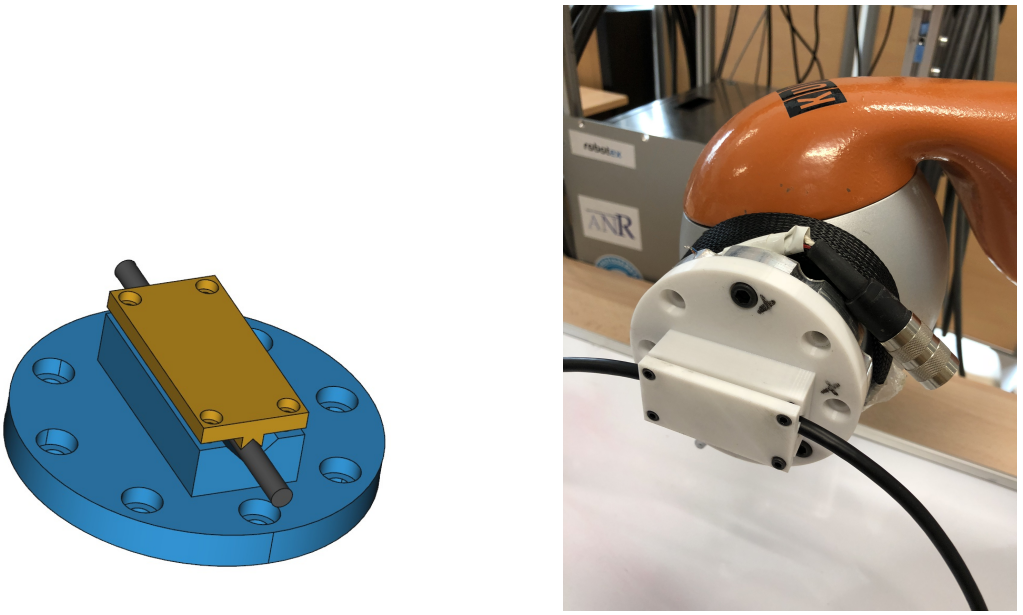
## 4.5 Robot Experiments

In this section, we introduce our setup for performing cable shaping tasks. Then, we briefly present the image processing techniques for extracting the cable from the image. In the end, we show and discuss the results of the experiments.

### 4.5.1 Hardware Setup

We use the BAZAR robot [Cherubini et al., 2019] equipped with two lightweight KUKA LWR IV robots for the dual arm setup (see Fig. 4.6).

We design two grippers to ensure the manipulation of the cable without slipping (see Fig. 4.5). In the left figure, the blue gear is attached to the robot. The black cylinder represents the cable, and the yellow gear fastens the cable rigidly.



(a) Cable gripper mechanical design.

(b) Cable gripper mounted on the robot.

Figure 4.5: The cable gripper.

The cable is gripped by two end-effectors. An Allied Vision® AVT camera is placed perpendicular to the manipulation plane to track the shape of the cable during manipulation. The image resolution is $1936 \times 1456$. A Linux-based 64-bits PC processes the image at a frame rate of 33.3Hz.

The overall hardware setup is showed in Fig. 4.6.

To test the robustness of the algorithm, we prepare two cables with different thicknesses. The diameter of the thin cable is 5.88 millimeters, and the thick cable is 7.66 millimeters. Under the same inputs from the robot, the two cables result in slightly different shapes.

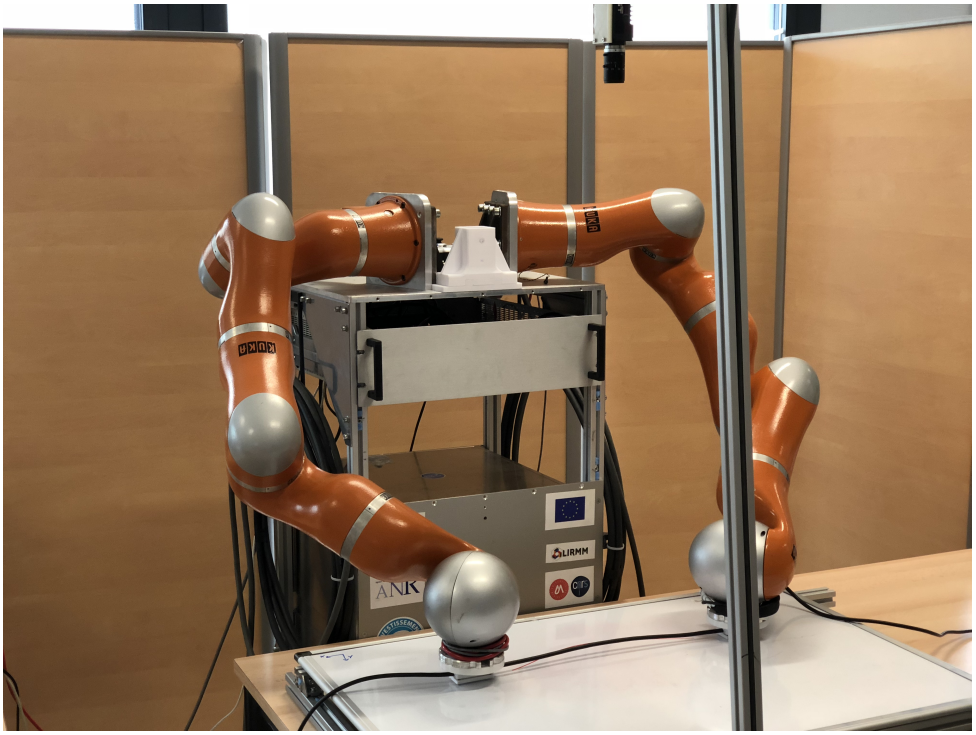


Figure 4.6: The hardware setup for cable manipulation.

### 4.5.2 Image Processing

We simplify the environment by putting a whiteboard on the table to serve as the 2D plane and using black cables for the manipulation. The color image is converted into a grayscale image. Then by thresholding the grayscale image, we can segment the cable from the image. The target shape is drawn on the whiteboard with red color markers. To detect the target shape, we transform the image representation from Red-Green-Blue (RGB) to Hue-Saturation-Value (HSV) and then apply thresholding to segment the red shape from the image. Figure 4.7 shows the image processing under experimental conditions. We use OpenCV for implementing the vision algorithms.

(a) Color image. (b) Cable segmentation. (c) Target shape.

Figure 4.7: Image processing.

### 4.5.3 Experiments

We calculate the target shape feature vector $\boldsymbol{s}^*$ from the binary image of the target shape (Fig. 4.7c) at the start of the control scheme. A camera continuously observes the shape

of the cable. The feature vector of the cable is constantly updated to compute the local deformation model and the control inputs. We choose $N = 2$ components of the Fourier series in (4.2). This value is chosen for the reasons:

1. Cable shapes reachable by the dual-arm are not complex, $N = 2$ can produce a reasonable approximation of the shape while $N = 1$ sometimes result in a bad approximation.

2. A Larger $N$ will result in more elements in the feature vector of the shape. Since the inputs from the dual arm robot are always six, when $N = 1$, the dimension of the feature vector is $4N + 2 = 6$, making the system fully actuated. When $N > 1$, with bigger $N$, the system becomes more and more underactuated, increasing the possibility of control being attracted to a local minimum.

3. Increasing $N$ will add in complexity of computation.

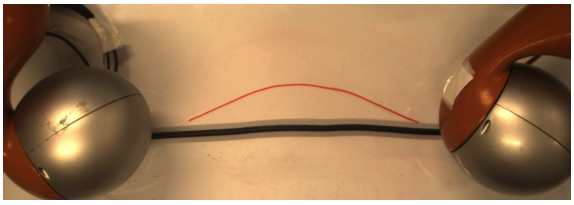
The following evaluation metric is designed to monitor the performance of the controller:

$$\Phi = \frac{||\boldsymbol{s}^* - \boldsymbol{s}||}{||\boldsymbol{s}^*||}.$$

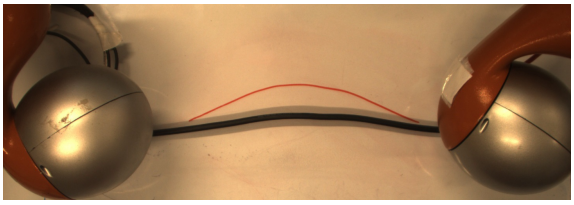
Lower $\Phi$ indicates better matches between the cable shape and the target shape.

To initialize $\boldsymbol{L}$ estimation, the robot executes a small motion that excites all components in $\delta\boldsymbol{r}$, so an initial estimation of $\boldsymbol{L}$ can be computed.

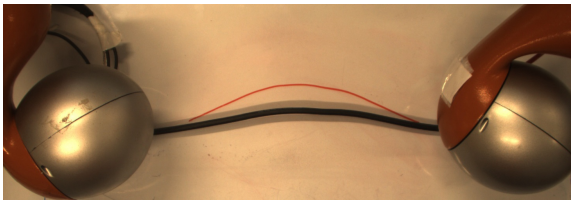
We test the algorithm on the thick cable with two different shapes. Figure 4.8 and 4.9 depict the experiments. The red color line is the target shape. In each figure, the initial and final shape of the cable are presented in (a) and (d), respectively. Figure (b) and (c) show the intermediate shapes while reaching the final shape.



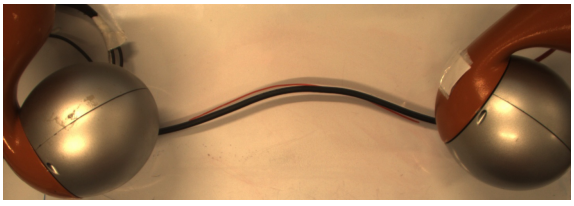
(a) Initial shape



(b) Intermediate shape 1

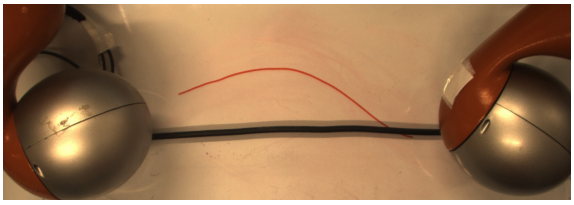


(c) Intermediate shape 2

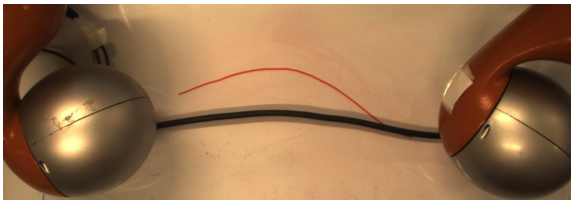


(d) Final shape

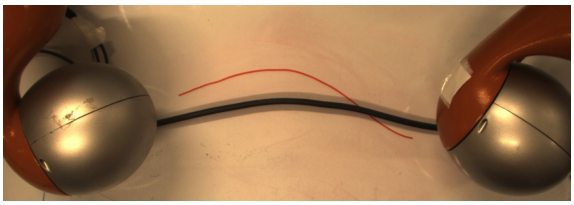
Figure 4.8: Experiment 1 - Thick cable.



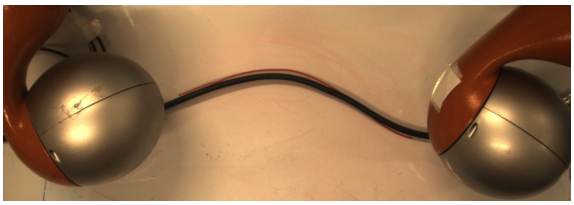
(a) Initial shape



(b) Intermediate shape 1



(c) Intermediate shape 2



(d) Final shape

Figure 4.9: Experiment 2 - Thick cable.

Figure 4.10 and 4.11 show the evolution of the performance metric during the manipulation. The general trend of the performance metric is decreasing, and the final shape overlaps the target shape, as shown in Figure 4.8 and 4.9.

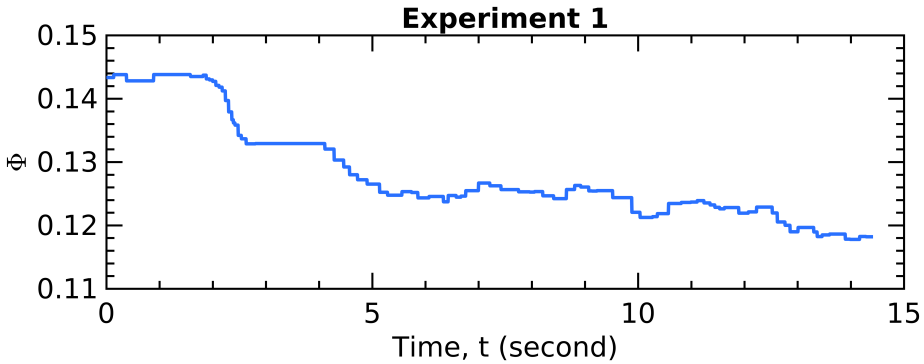



Figure 4.10: Performance metric - Experiment 1.

To further test the robustness of our algorithm, we test it on a thinner cable. Figure 4.12 shows the shape servoing steps. Fig. 4.13 depicts the evolution of the performance metric.

In the subsequent experiment, we change the initial configuration of the cable. Figure 4.14 shows the manipulation results. It proves that the algorithm is invariant to the initial configuration of the cable.

The shape servoing tasks are performed with no knowledge of the deformation characteristic of the cable. The algorithm is verified on two distinct cables to show robustness. As the interaction matrices directly relates the robot motion to the shape changes on the image, our method does not require camera calibration, i.e., for knowing the intrinsic and extrinsic parameters of the camera.

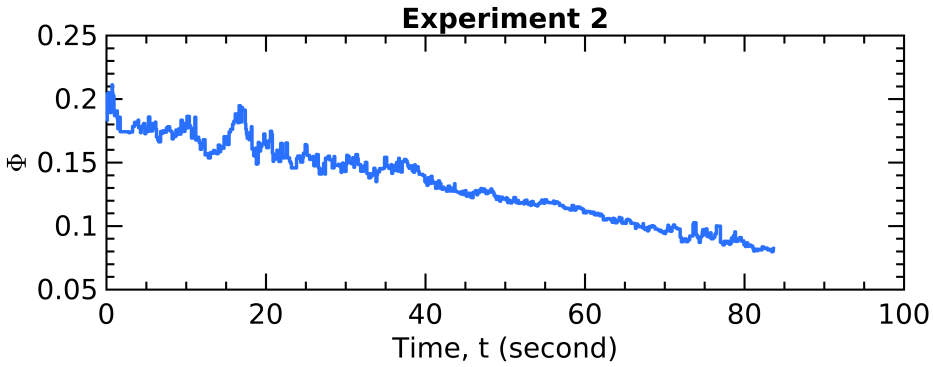
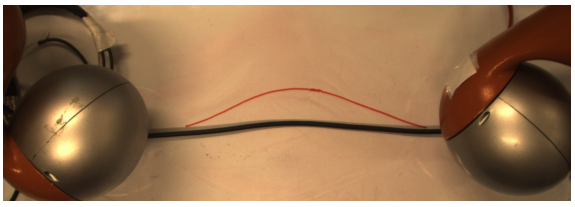
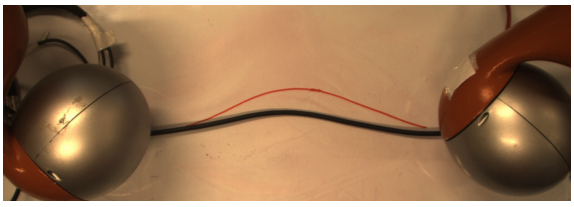


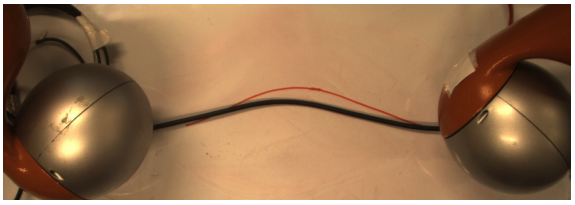

Figure 4.11: Performance metric - Experiment 2.



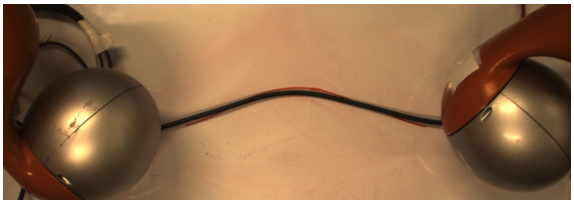
(a) Initial shape



(b) Intermediate shape 1



(c) Intermediate shape 2



(d) Final shape

Figure 4.12: Experiment 3 - Thin cable.

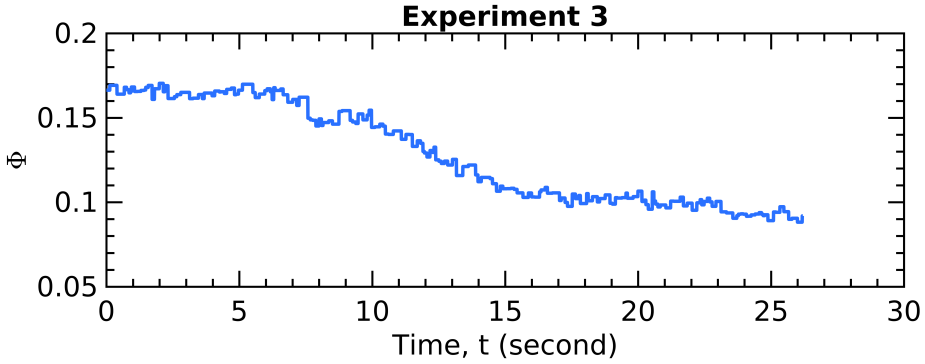
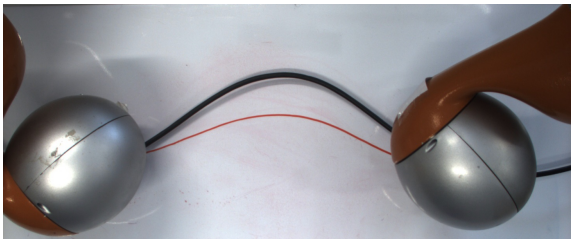
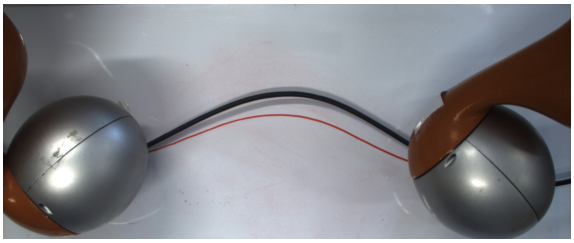


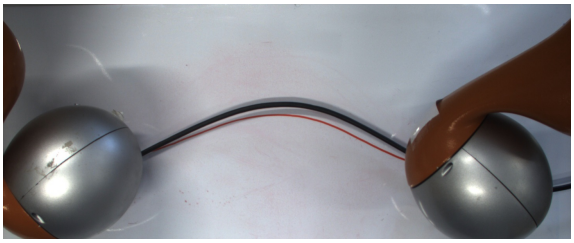

Figure 4.13: Performance metric - Experiment 3.



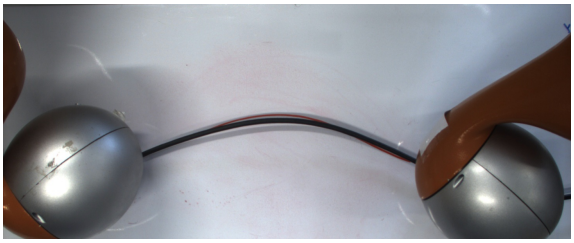
(a) Initial shape



(b) Intermediate shape 1



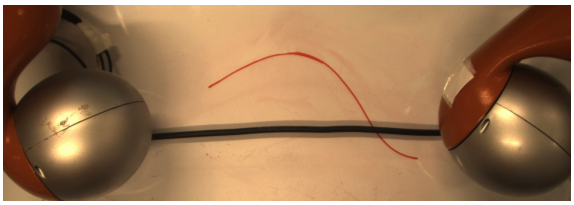
(c) Intermediate shape 2



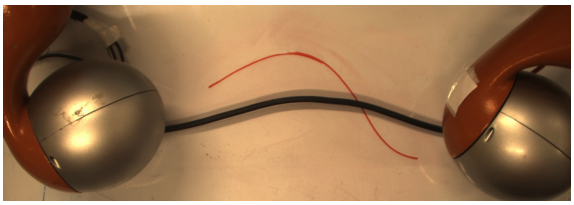
(d) Final shape

Figure 4.14: Experiment 4 - Thin cable, different starting configuration.

Fig. 4.15 shows one failure in reaching the target shape. Fig. 4.15a and 4.15b show the initial shape and final shape of the cable, respectively, and the red line depicts the target shape. The robot goes back and forth near the final shape and is not able to reach the target shape.



(a) Initial shape



(b) Final shape

Figure 4.15: Experiment 5 - Unreachable target.

The full video of the experiment can be found at `http://y2u.be/DPl_d7lbL84`.

The failure case may be due to under-actuation. With six inputs from the robot and ten parameters to control, the system is underactuated. Also, the deformation model is locally approximated, so if the target shape deviates too much from the initial shape, without path planning, the control will more likely converge to a local minimum. The approximation accuracy can also affect the convergence.

## 4.6 Conclusion

In this starting work on deformable object manipulation, we adopt the model-free method proposed in [Navarro-Alarcon and Liu, 2017] and extend the method to a dual arm robot

on the open contours. By introducing rotation and the second arm in the experiment, the robot is able to perform cable deformation tasks as humans.

The Fourier-based parameterization results in an unnecessary large dimension of the feature vector, which makes the control problem under-actuated. Due to the lack of a global deformation model, we cannot prove the global convergence of the algorithm, nor can we tell if a certain shape is reachable or not.

With the lesson learned from this starting research, in the next chapter, we present a novel and generalized model-free solution for shape servoing of deformable and rigid objects alike. As the subsequent and more thorough research in this direction, we will present a more detailed theory and back it up with comprehensive experiments.

Chapter 5

# Subspace-based Object Manipulation

Humans are capable of manipulating both rigid and deformable objects. However, robotic researchers tend to consider the manipulation of these two classes of objects as separate problems. This chapter presents our efforts in formulating a generalized framework for vision-based manipulation of both rigid and deformable objects, which does not require prior knowledge of the object's mechanical properties.

In classical visual servoing literature [Chaumette and Hutchinson, 2006b], the vector $\boldsymbol{s}$ denotes the set of features selected to represent the object in the image. These features represent both the object's pose and its shape. We denote the process of selecting $\boldsymbol{s}$ as *parameterization*. The aim of visual servoing is to minimize, through robot motion, the feedback error $\boldsymbol{e} = \boldsymbol{s}^* - \boldsymbol{s}$ between the desired $\boldsymbol{s}^*$ and the current (i.e., measured) feature $\boldsymbol{s}$.

One of the initial works on vision-based manipulation of deformable objects is presented in [Inoue, 1984] to solve a knotting problem by a topological model. Smith et al. developed a relative elasticity model, such that vision can be utilized without a physical model for the manipulation task [Smith et al., 1996]. A classical model-free approach in manipulating deformable objects is developed in [Berenson, 2013]. More recent research [Lagneau et al., 2020] proposes a method for online estimation of the deformation Jacobian, based on weighted least square minimization with a sliding window. In [Navarro-Alarcon et al., 2014] and [Navarro-Alarcon and Liu, 2017], the vision-based deformable objects manipulation is termed as *shape servoing*. An expository paper on the topic is available in [Navarro-Alarcon et al., 2019]. A recent work on shape servoing of plastic material was presented in [Cherubini et al., 2020].

For *shape servoing*, commonly selected features are curvatures [Navarro-Alarcon et al., 2014], points [Wang et al., 2018] and angles [Navarro-Alarcon and Liu, 2013]. Laranjeira et al. proposed a catenary-based feature for tethered management on wheeled and underwater robots [Laranjeira et al., 2017, Laranjeira et al., 2020]. A more general feature vector is that containing the Fourier coefficients of the object contour [Navarro-Alarcon and Liu, 2017] and [Zhu et al., 2018]. Yet, all these approaches require the user to specify a model, e.g., the object geometry [Wang et al., 2018, Navarro-Alarcon and Liu, 2013, Navarro-Alarcon et al., 2014] or a function [Laranjeira et al., 2017, Navarro-Alarcon and Liu, 2017, Zhu et al., 2018] for selecting the feature. Alternative data-driven (hence, modeling-free) approaches rely on machine learning. Nair et al. combine learning and

visual feedback to manipulate ropes in [Nair et al., 2017]. The authors of [Hu et al., 2019] employ deep neural networks to manipulate deformable objects given their 3D point cloud. All these methods rely on (deep) connectionists models that invariably require training through an extensive data set. The collected data has to be diverse enough to generalize the model learnt by this type of networks. Furthermore, the above-mentioned methods focus only on deformable object manipulation.

As for *pose control of rigid objects*, the trend in visual servoing is to find features which are independent from the object characteristics. Following this trend, [Chaumette, 2004] proposes the use of image moments. More recently, researchers have proposed direct visual servoing (DVS) methods, which eliminate the need for user-defined features and for the related image processing procedures. The pioneer DVS works [Collewet et al., 2008, Collewet and Marchand, 2011] propose using the whole image luminance to control the robot, leading to "photometric" visual servoing. Bakthavatchalam et al. join the two ideas by introducing photometric moments [Bakthavatchalam et al., 2013]. A subspace method [Marchand, 2019] can further enhance the convergence of photometric visual servoing, via Principal Component Analysis (PCA). This method was first introduced for visual servoing in [Nayar et al., 1996]. In that work, using an eye-in-hand setup, the image was compressed to obtain a low-dimensional vector for controlling the robot to a desired pose. Similarly, the authors of [Deguchi and Noguchi, 1996] transformed the image into a lower dimensional hyper surface, to control the robot position via in-hand camera feedback. However, DVS generally considers rigid and static scenes, where the robot controls the motion of the camera (eye-in-hand setup) to change only the image viewpoint, and not the environment. This setup avoids breaking the Lambertian hypotheses, which are needed for the methods to be applicable. For this reason, to our knowledge, DVS was never applied to object manipulation, since changes in the pose and/or shape of the object would break the Lambertian assumption.

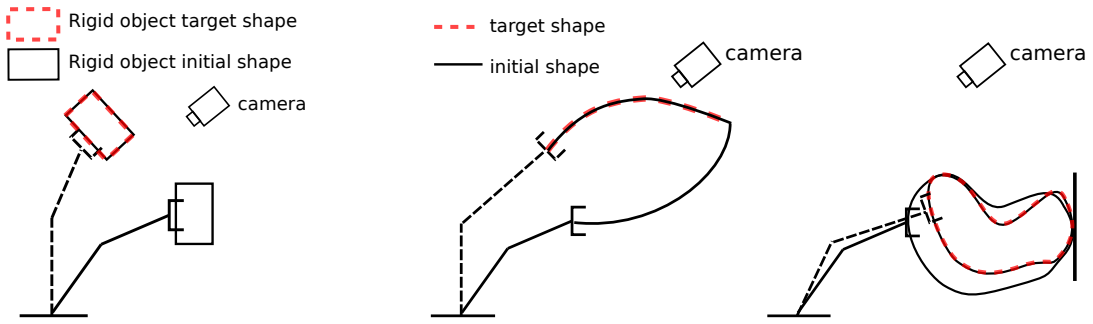
Compared with above-mentioned works, our approach has the following original contributions:

1. We introduce a new compact feedback feature vector – based on PCA of sampled 2D contours – which can be used for both deformable and rigid object manipulation.

2. We exploit the linear properties of PCA and of the local interaction matrix, to initialize our algorithm with little data – the same data for feature vector extraction and for interaction matrix estimation.

3. We report experiments using the same framework to manipulate objects with different unknown geometric and mechanical properties.

In a nutshell, we propose an efficient data-driven approach that allows manipulation of an object regardless of its deformation characteristics. To the best of authors' knowledge, there has not been any similar framework proposed before.




(a) Rigid objects (b) Deformable objects

Figure 5.1: Vision-based manipulation of rigid and deformable objects. For rigid objects (left): control pose (translation and rotation). For deformable objects (right): control the pose, and also shape.

The chapter is organized as follows. Sect. 5.1 presents the problem. Sect. 5.2 outlines the framework. Sect. 5.3 elaborates on the methods. In Sect. 5.4, we analyze and verify the methods by numerical simulations. Then, Sect. 5.5 presents the robotic experiments and we conclude in Sect. 5.6.

## 5.1 Problem Statement

In this work, we aim at solving object manipulation tasks with visual feedback. We rely on the following hypotheses:

- The shape and pose of the object are represented by its 2-D contour on the image as seen from a camera fixed in the robot workspace (eye-to-hand setup). We denote this contour as
$$\boldsymbol{c} = [\boldsymbol{p}_1 \ \cdots \ \boldsymbol{p}_K]^T \in \mathbb{R}^{2K}, \tag{5.1}$$
where $\boldsymbol{p}_j = [u_j \ v_j] \in \mathbb{I}$ denotes the $j$th pixel of the contour in the image $\mathbb{I}$.

- The contour is always entirely visible in the scene and there are no occlusions.

- One of the robot's end-effectors holds one point of the object (we consider the grasping problem to be already solved). At each control iteration $i$, its pose is $\boldsymbol{r}_i \in \mathbb{SE}(3)$, and it can execute motion commands $\delta\boldsymbol{r}_i \in \mathbb{SE}(3)$ that drive the robot as $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \delta\boldsymbol{r}_i$.

**Problem Statement.** Given a desired shape of the object, represented by its contour $\boldsymbol{c}^*$, we aim at designing a controller that generates a sequence of robot motions $\delta\boldsymbol{r}$ to drive the initial contour to the desired one, by relying on visual feedback.

The controller should work without any knowledge of the object physical characteristics (i.e., for both rigid and deformable objects). Typically, since rigid and deformable objects behave differently during manipulation, we set the following manipulation goals:

- Rigid objects: move them to a desired pose (see Fig. 5.1a).
- Deformable objects: move them to a desired pose with a desired shape (see Fig. 5.1b).

## 5.2 Framework Overview

In this section, we present an overview of the proposed approach motivated by the problem analysis.

### 5.2.1 Problem Analysis

We can work directly on the object shape space by selecting the contour as the feature vector $\boldsymbol{s} \equiv \boldsymbol{c} \in \mathbb{R}^{2K}$. However, this will result in an unnecessarily large dimension of the feature vector (e.g., if $K = 50$, $\boldsymbol{s}$ has 100 components). The high dimensional feature vector increases the computation demand and complicates the control due to the high under-actuation of the system. Therefore, instead of working directly on shape space, we work on a feature vector space that has reduced dimensions. For that, we split the problem into two sub-problems: *parameterization* and *control*, see Fig. 5.2.

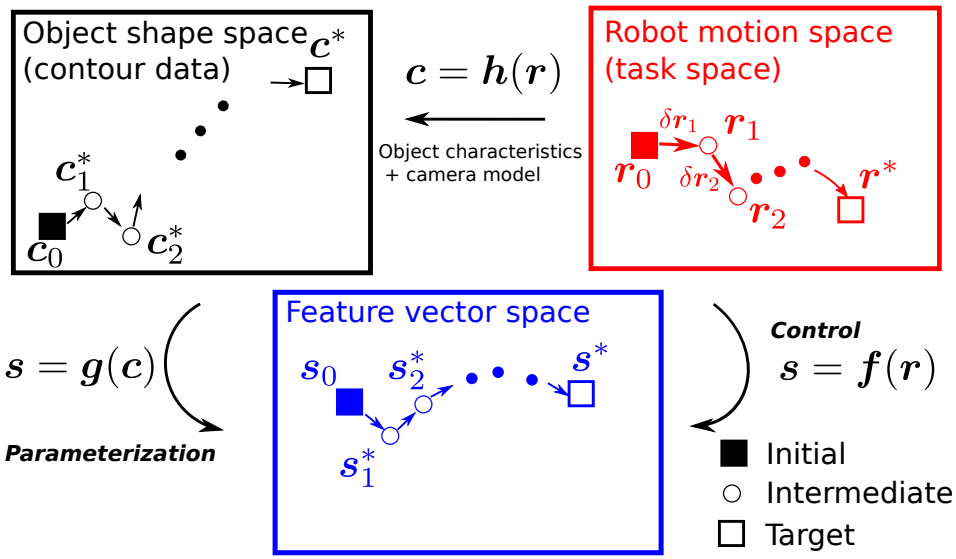



Figure 5.2: Graphic representation of the vision-based manipulation problem, with its two sub-problems, *parameterization* and *control*.

*Parameterization* consists in representing the shape contour via a compact feature vector $\boldsymbol{s} \in \mathbb{R}^k$, such that $k << 2K$. We denote this representation as $\boldsymbol{s} = \boldsymbol{g}(\boldsymbol{c})$.

*Control* consists in computing robot motions $\delta\boldsymbol{r}_1, \delta\boldsymbol{r}_2, \ldots$, so that the object's representation $\boldsymbol{s}$ converges to a desired target $\boldsymbol{s}^*$. *Control* can be broken down to solving the optimization problem:

$$\boldsymbol{r}^* = \arg\min_{\boldsymbol{r}}(\boldsymbol{f}(\boldsymbol{r}) - \boldsymbol{s}^*) \tag{5.2}$$

where $\boldsymbol{s} = \boldsymbol{f}(\boldsymbol{r})$ denotes the mapping between robot pose and feature vector, which is assumed to be smooth and often nonlinear. If we know the analytic solution to $\boldsymbol{f}(\boldsymbol{r})$, we can solve (5.2) and obtain the target shape in a single iteration by commanding $\boldsymbol{r}^*$. However, the full mapping $\boldsymbol{f}(\boldsymbol{r})$ is the result of two subsequent mappings. A mapping $\boldsymbol{c} = \boldsymbol{h}(\boldsymbol{r})$ exists between the robot pose $\boldsymbol{r}$ and the resulting contour $\boldsymbol{c}$. This, combined with the *parameterization* mapping above yields: $\boldsymbol{f}(\boldsymbol{r}) = \boldsymbol{g}(\boldsymbol{h}(\boldsymbol{r}))$. To solve for $\boldsymbol{f}(\boldsymbol{r})$, one needs to have the analytic expressions of both $\boldsymbol{h}$ and $\boldsymbol{g}$. While the latter comes from the *parameterization*, the former ($\boldsymbol{h}$) is difficult to obtain, since it encompasses both the object characteristics and the camera projection model. For different objects and camera poses, we expect different properties and camera parameters, therefore a different $\boldsymbol{h}$. Even for $\boldsymbol{g}$, we would like our framework to automatically extract the feature vector without explicit human definition. Therefore, unlike traditional approaches where $\boldsymbol{g}$ is user-defined and known, in our framework, the robot has to infer $\boldsymbol{g}$ from sensor data, which in return makes deriving $\boldsymbol{g}$ difficult.

A solution to this problem is to approximate the full mapping $\boldsymbol{f}(\boldsymbol{r})$ from sensor observations. Classic data-driven approaches typically require a long training phase to collect vast and diverse data for approximating $\boldsymbol{f}(\boldsymbol{r})$. In some cases (robotics surgery for instance), it is not possible to collect such data beforehand. Moreover, if the object changes, new data has to be collected to retrain the model, leading to a cumbersome process. In this work instead, we aim at doing the data collection online, with minimum initialization.

Thus, instead of estimating the full nonlinear mapping $\boldsymbol{f}(\boldsymbol{r})$, we divide it into piecewise linear models [Sang and Tao, 2012] at successive equilibrium points. These models are considered time invariant in the neighbourhood of the equilibrium points. We then compute the control law for each linear model and apply it to the robot end-effector.

### 5.2.2 Proposed Methods

Given a target shape $\boldsymbol{c}^*$, we define an intermediate local target $\boldsymbol{c}_i^*$ at each $i = 1, 2, \ldots$ (see Fig. 5.2). At the $i^{\text{th}}$ instance, the robot autonomously generates a local mapping $\boldsymbol{g}_i$ to produce the feature vector $\boldsymbol{s}_i = \boldsymbol{g}_i(\boldsymbol{c}_i)$. The robot then finds the local mapping $\boldsymbol{s}_i = \boldsymbol{f}_i(\boldsymbol{r}_i)$ online.

Consider at the current time instant $i$, the shape $\boldsymbol{c}_i$, the intermediate target $\boldsymbol{c}_i^*$ and

the local parameterization $\boldsymbol{g}_i$, we can transform shape data into a feature vector by:

$$\boldsymbol{s}_i = \boldsymbol{g}_i(\boldsymbol{c}_i),\ \boldsymbol{s}_i^* = \boldsymbol{g}_i(\boldsymbol{c}_i^*). \tag{5.3}$$

The linearized version of $\boldsymbol{s} = \boldsymbol{f}(\boldsymbol{r})$ centered at $(\boldsymbol{s}_i, \boldsymbol{r}_i)$ is then:

$$\delta\boldsymbol{s}_i = \boldsymbol{L}_i \delta\boldsymbol{r}_i, \tag{5.4}$$

with

$$\begin{aligned} \boldsymbol{L}_i &= \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{r}} \mid_{\boldsymbol{r}=\boldsymbol{r}_i}, \\ \delta\boldsymbol{s}_i &= \boldsymbol{s}_{i+1} - \boldsymbol{s}_i, \\ \delta\boldsymbol{r}_i &= \boldsymbol{r}_{i+1} - \boldsymbol{r}_i. \end{aligned} \tag{5.5}$$

The matrix $\boldsymbol{L}_i$ represents a local mapping, referred to as the interaction matrix in the visual servoing literature [Chaumette and Hutchinson, 2006b]. If $\boldsymbol{L}_i$ can be estimated online at each iteration $i$, then, we can design one-step control laws to drive $\boldsymbol{s}_i$ towards $\boldsymbol{s}_i^*$.

After the robot has executed the motion command $\delta\boldsymbol{r}_i$, we update the next target to be $\boldsymbol{s}_{i+1}^*$, and so on, until it reaches the final (desired) target $\boldsymbol{s}^*$. Although the validity region of this local linear mapping is much smaller than that of the original nonlinear mapping, it results in an online training with less data and reduced computational demand.

In the following section, we detail our proposed approach.

## 5.3 Methods

Figure 5.3 shows the building blocks for the overall framework. In this section, we focus on the red dashed part of the diagram. We will elaborate on each red block in the subsequent subsections. The blue block represents the image processing pipeline that will be discussed in Sect. 5.5.1.

### 5.3.1 Feature Vector Extraction

There are many ways for parameterizing $\boldsymbol{c}$ to reduce its dimension. Since the interaction matrix in (5.4) represents a *linear* mapping for the control, to be consistent, we look for a *linear* transformation for the parameterization.

One of the prominent linear dimension reduction methods is Principal Component Analysis (PCA). PCA finds a new orthogonal basis for high-dimensional data. This enables projection of the data to lower dimension with the minimal sum of squared residuals. We apply PCA to map $\boldsymbol{c} \in \mathbb{R}^{2K}$ to $\boldsymbol{s} \in \mathbb{R}^k$.

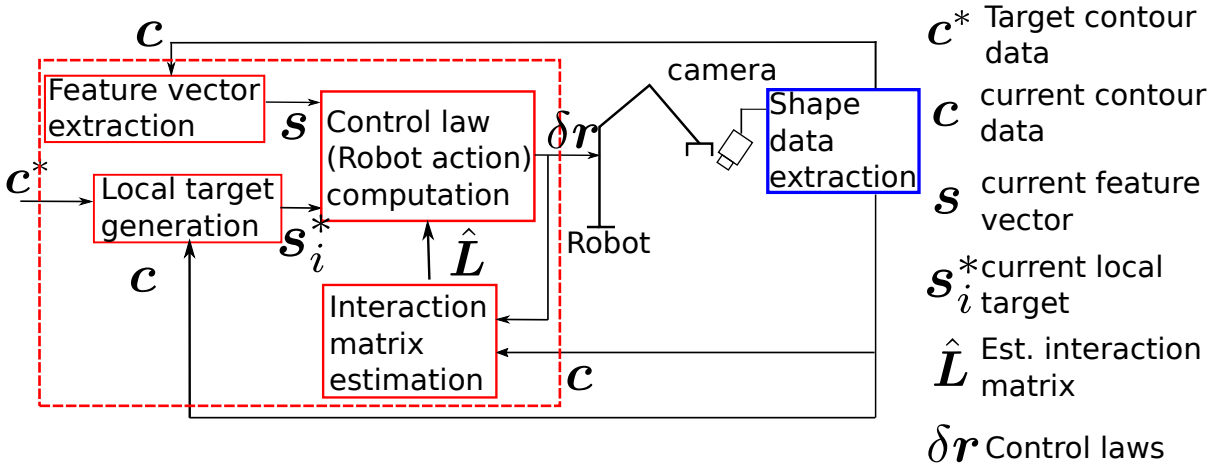



Figure 5.3: The block diagram that represents the overall framework.

To find the projection, we collect $M$ images with different shapes of the object and construct the data matrix $\mathbf{\Gamma} = [\boldsymbol{c}_1 \ \boldsymbol{c}_2 \ \cdots \ \boldsymbol{c}_M] \in \mathbb{R}^{2K \times M}$. Then, we shift the columns of $\mathbf{\Gamma}$ by the mean contour $\bar{\boldsymbol{c}} = \sum_{i=1}^{M} \boldsymbol{c}_i / M$:

$$\bar{\mathbf{\Gamma}} = [\boldsymbol{c}_1 - \bar{\boldsymbol{c}} \quad \boldsymbol{c}_2 - \bar{\boldsymbol{c}} \quad \cdots \ \boldsymbol{c}_M - \bar{\boldsymbol{c}}] \in \mathbb{R}^{2K \times M}. \tag{5.6}$$

We then compute the covariance matrix $\boldsymbol{C} = \bar{\mathbf{\Gamma}}^T \bar{\mathbf{\Gamma}}$, and apply Singular Value Decomposition (SVD) to it:

$$\boldsymbol{C} = \boldsymbol{U} \mathbf{\Sigma} \boldsymbol{V}^T. \tag{5.7}$$

Once we have obtained the eigenvector matrix $\boldsymbol{U} \in \mathbb{R}^{2K \times 2K}$, we can move on to select the first $k$ columns of $\boldsymbol{U}$ denoted by $\boldsymbol{U}(k) \in \mathbb{R}^{2K \times k}$. Then, the $2K$-dimensional contour $\boldsymbol{c}$ can be projected into a smaller $k$-dimensional feature vector $\boldsymbol{s}$ as:

$$\boldsymbol{s} = \boldsymbol{U}^T(k)(\boldsymbol{c} - \bar{\boldsymbol{c}}) \in \mathbb{R}^k. \tag{5.8}$$

To assess the quality of this projection, we can compute the *explained variance* using the eigenvalue matrix $\mathbf{\Sigma} \in \mathbb{R}^{2K \times 2K}$ in (5.7). Denoting the diagonal entries of $\mathbf{\Sigma}$ as $\sigma_1, \cdots, \sigma_{2K}$, the explained variance of the first $k$ components is:

$$\Upsilon(k) = \frac{\sum_{j=1}^{k} \sigma_j}{\sum_{j=1}^{2K} \sigma_j}. \tag{5.9}$$

where $\Upsilon$ is a scalar between 0 and 1 (since $\sigma_j > 0, \forall j$), indicating to what extent the $k$ components represent the original data (a larger $\Upsilon$ suggests a better representation).

At this stage, it should be clear for the reader that there is a crucial trade-off between choosing a low or high value for the number of features, $k$. A low $k$ will ease controllability (given the limited robot inputs), whereas a high $k$ will improve the data representation

(by increasing $\Upsilon$). In the next section, we explain how to select the best value of $k$.

### 5.3.2 Selection of the Feature Dimension $k$

Feature vector $\boldsymbol{s} \in \mathbb{R}^k$ in (5.8) lies in the new orthonormal basis represented by the columns of $\boldsymbol{V}$ in (5.7). Therefore, both $\boldsymbol{s}$ and its variation $\delta\boldsymbol{s}$ span a space of dimension $k$. Similarly, $\boldsymbol{r}$ and $\delta\boldsymbol{r}$ span a space of dimension $n$.

From (5.4), we know that at each iteration, $\delta\boldsymbol{s} \in \mathbb{R}^k$ is the result of a *constant* linear mapping ($\boldsymbol{L}$) on $\delta\boldsymbol{r} \in \mathbb{R}^n$. For an arbitrary number $p$ of samples of small robot motions $\delta\boldsymbol{r}$: $\boldsymbol{R} = \begin{bmatrix}\delta\boldsymbol{r}_1 \cdots \delta\boldsymbol{r}_p\end{bmatrix}$ such that the mapping $\boldsymbol{L}$ stays constant, we have $\text{rank}(\boldsymbol{R}) \leq n$. Using (5.4) we obtain

$$\boldsymbol{S} = \boldsymbol{L}\boldsymbol{R} \tag{5.10}$$

such that $\boldsymbol{S} = \begin{bmatrix}\delta\boldsymbol{s}_1 \cdots \delta\boldsymbol{s}_p.\end{bmatrix}$. Each column of $\boldsymbol{S}$ is the result of corresponding robot motion. To find the relationship between $k$ and $n$, we introduce the following lemma [Strang, 1993]:

**Lemma 1.** $\text{rank}(\boldsymbol{A}\boldsymbol{B}) \leq \min(\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B}))$.

Applying *Lemma 1* to (5.10):

$$\begin{aligned}\text{rank}(\boldsymbol{S}) &\leq \min(\text{rank}(\boldsymbol{L}), \text{rank}(\boldsymbol{R})) \\ &\leq \text{rank}(\boldsymbol{R}) \leq n.\end{aligned} \tag{5.11}$$

Since $\delta\boldsymbol{s} \in \mathbb{R}^k$, the maximum dimension of $\text{rank}(\boldsymbol{S})$ is $k$: $\text{rank}(\boldsymbol{S}) \leq k$. According to (5.11), $k \leq n$. Since a larger $k$ yields a better approximation of the real shape data, the value of $k$ should be chosen as large as possible considering the condition $k \leq n$. Therefore we choose $k = n$.

### 5.3.3 Local Target Generation

Let us now explain how we generate a local desired contour $\boldsymbol{c}_i^*$ given current contour $\boldsymbol{c}_i$ and final desired contour $\boldsymbol{c}^*$ (Algorithm 1). The overall shape error is given by:

$$\boldsymbol{c_e} = \boldsymbol{c}^* - \boldsymbol{c}_i. \tag{5.12}$$

We define the intermediate desired contour as:

$$\boldsymbol{c}_i^* = \boldsymbol{c}_i + \frac{1}{\eta}\boldsymbol{c_e}, \tag{5.13}$$

with $\eta = 1, 2, \ldots$ an integer that ensures that $\boldsymbol{c}_i^*$ is a "good" local target for $\boldsymbol{c}_i$ (i.e., the two are similar). This means that if we project the intermediate local data using $\boldsymbol{U}_i \in \mathbb{R}^{2K\times 2K}$ (note that we are using the full projection matrix and not just the first $k$ columns), the

feature vector $\boldsymbol{s}_i^* = \boldsymbol{U}_i(\boldsymbol{c}_i^* - \bar{\boldsymbol{c}}) \in \mathbb{R}^{2K}$ should fulfil:

$$\Psi(k) = \frac{\sum_{j=1}^{k} \left| s_{i,j}^* \right|}{\sum_{j=1}^{2K} \left| s_{i,j}^* \right|} \geq \epsilon, \tag{5.14}$$

with $\epsilon \in ]0;1[$ a threshold and $s_{i,j}^*$ the $j$-th component of $\boldsymbol{s}_i^*$.

Algorithm 1 outlines the steps for computing the local intermediate targets, so that:

- they are near the final target,
- the corresponding feature vector can be extracted with the current learned projection matrix.

**Remark 1.** The reachability of a local target can only be verified with a global deformation model which we want to avoid identifying in our methods. We will further discuss this issue in the Conclusion (Sect. 5.6).

**Algorithm 1** Local target generation

```
localTargetFound = false
Ψ_0 = 0
η = 1
while not localTargetFound do
    c_i^* = c_i + (1/η)(c^* − c_i)
    s_i^* = U_i c_i^*
    Ψ_η = Σ_{j=1}^{k} |s_{i,j}^*| / Σ_{j=1}^{2K} |s_{i,j}^*|
    if Ψ_η ≥ ε or Ψ_η < Ψ_{η−1} then
        localTargetFound = true
    η = η + 1
```

### 5.3.4 Interaction Matrix Estimation

Let us consider the current contour $\boldsymbol{c}_i$ and the local target $\boldsymbol{c}_i^*$. In this section, we show how we can implement the PCA and model estimation together and online. We denote the robot motions and corresponding object contours over the last $M$ iterations (prior to iteration $i \geq M$) as:

$$\begin{aligned} \boldsymbol{\Delta R}_i &= \left[ \delta \boldsymbol{r}_{i-M+1}\ \delta \boldsymbol{r}_{i-M+2} \cdots \delta \boldsymbol{r}_i \right] \in \mathbb{R}^{n \times M} \\ \boldsymbol{\Gamma}_i &= \left[ \boldsymbol{c}_{i-M}\ \boldsymbol{c}_{i-M+1}\ \boldsymbol{c}_{i-M+2} \cdots \boldsymbol{c}_i \right] \in \mathbb{R}^{2K \times (M+1)} \end{aligned} \tag{5.15}$$

By selecting $k = n$, we compute the projection matrix $\boldsymbol{U}_i(n) \in \mathbb{R}^{2K \times n}$, from $\boldsymbol{\Gamma}_i$ and $\bar{\boldsymbol{c}}_i$ via (5.6) and (5.7). Then, using $\boldsymbol{U}_i$, we project current contour $\boldsymbol{c}_i$, desired contour $\boldsymbol{c}_i^*$ and

shape matrix $\boldsymbol{\Gamma}_i$:

$$\begin{aligned}
\boldsymbol{s}_i &= \boldsymbol{U}_i(n)^T(\boldsymbol{c}_i - \bar{\boldsymbol{c}}_i) \in \mathbb{R}^n, \\
\boldsymbol{s}_i^* &= \boldsymbol{U}_i(n)^T(\boldsymbol{c}_i^* - \bar{\boldsymbol{c}}_i) \in \mathbb{R}^n, \\
\boldsymbol{S}_i &= \boldsymbol{U}_i(n)^T\bar{\boldsymbol{\Gamma}}_i = \begin{bmatrix} \boldsymbol{s}_{i-M} & \boldsymbol{s}_{i-M+1} & \cdots & \boldsymbol{s}_i \end{bmatrix} \in \mathbb{R}^{n\times(M+1)}.
\end{aligned} \tag{5.16}$$

In (5.16), $\bar{\boldsymbol{\Gamma}}_i$ is normalized by $\bar{\boldsymbol{c}}_i$ as in (5.6). We can then compute $\boldsymbol{\Delta S}_i$ from (5.16), by subtracting consecutive columns of $\boldsymbol{S}_i$:

$$\boldsymbol{\Delta S}_i = \begin{bmatrix} \delta\boldsymbol{s}_{i-M+1} & \delta\boldsymbol{s}_{i-M+2} \cdots \delta\boldsymbol{s}_i \end{bmatrix} \in \mathbb{R}^{n\times M}. \tag{5.17}$$

Using $\boldsymbol{\Delta S}_i \in \mathbb{R}^{n\times M}$ and $\boldsymbol{\Delta R}_i \in \mathbb{R}^{n\times M}$ we can now estimate the local interaction matrix $\boldsymbol{L}_i \in \mathbb{R}^{n\times n}$ at iteration $i$. We assume that near this iteration, the system remains linear and time invariant: $\boldsymbol{L}_i$ is constant. Using the local linear model (5.4), we can write the following:

$$\boldsymbol{\Delta S}_i = \boldsymbol{L}_i\boldsymbol{\Delta R}_i. \tag{5.18}$$

Our goal then is to solve for $\boldsymbol{L}_i$, given $\boldsymbol{\Delta S}_i$ and $\boldsymbol{\Delta R}_i$. Note that this is an overdetermined linear system (with $n\times M$ equations for $n^2$ unknonwns). Let us consider $\boldsymbol{\Delta R}_i \in \mathbb{R}^{n\times M}$ has full row rank. Note this sufficiently implies $M \geq n$. With this prerequisite, $\text{rank}(\boldsymbol{\Delta R}_i) = n$. Therefore, $\text{rank}(\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T) = n$, and its inverse exists. We post multiply (5.18) by $\boldsymbol{R}_i^T$:

$$\boldsymbol{\Delta S}_i\boldsymbol{R}_i^T = \boldsymbol{L}_i\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T. \tag{5.19}$$

Then, since $\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T$ is invertible, the $\boldsymbol{L}_i$ that best fulfills (5.18) is:

$$\hat{\boldsymbol{L}}_i = \boldsymbol{\Delta S}_i\boldsymbol{\Delta R}_i^T(\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T)^{-1}. \tag{5.20}$$

This matrix is also the solution of the optimization problem

$$\min_{\boldsymbol{L}_i} \sum_{j=1}^{n} \left\| \delta\boldsymbol{\sigma}_j^T - \Delta\boldsymbol{R}_i^T\boldsymbol{l}_j^T \right\|^2, \tag{5.21}$$

where $\boldsymbol{\sigma}_j$ and $\boldsymbol{l}_j$ are respectively the $j^{th}$ row of $\Delta\boldsymbol{S}_i$ and $\boldsymbol{L}_i$. The detailed derivation is presented in the Appendix.

If the full row rank condition of $\boldsymbol{\Delta R}_i$ is not satisfied, $\text{rank}(\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T) < n$ and $\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T$ is singular. Then, instead of (5.20), we can use Tikhonov regularization:

$$\hat{\boldsymbol{L}}_i = \boldsymbol{\Delta S}_i\boldsymbol{\Delta R}_i^T(\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T + \lambda\boldsymbol{I})^{-1}. \tag{5.22}$$

Practically, this implies that one or more inputs motions do not appear in $\boldsymbol{\Delta R}_i$. Therefore, we cannot infer the relationship between these motions and the resulting feature

vector changes. In this case it is better to increase $M$ and obtain more data, so that $\boldsymbol{\Delta R}_i$ has full row rank.

Instead of computing the interaction matrix, it is also possible to directly compute its inverse, since this guarantees better control properties, as explained in [Lapresté et al., 2004]. With the same data, one can re-write (5.18) as:

$$\boldsymbol{L}_i^{\oplus} \boldsymbol{\Delta S}_i = \boldsymbol{\Delta R}_i. \tag{5.23}$$

We can also solve (5.23) with Tikhonov regularization:

$$\hat{\boldsymbol{L}}_i^{\oplus} = \boldsymbol{\Delta R}_i \boldsymbol{\Delta S}_i^T (\boldsymbol{\Delta S}_i \boldsymbol{\Delta S}_i^T + \lambda \boldsymbol{I})^{-1}. \tag{5.24}$$

### 5.3.5 Control Law and Stability Analysis

It is now possible to control the robot, using either of the following strategies:

$$\delta \boldsymbol{r}_i = -\alpha \hat{\boldsymbol{L}}_i^{\dagger} (\boldsymbol{s}_i - \boldsymbol{s}_i^*), \tag{5.25}$$

if one estimates the interaction matrix with (5.22), where $^{\dagger}$ denotes the pseudo-inverse, or:

$$\delta \boldsymbol{r}_i = -\alpha \hat{\boldsymbol{L}}_i^{\oplus} (\boldsymbol{s}_i - \boldsymbol{s}_i^*) \tag{5.26}$$

if one estimates the inverse of the interaction matrix with (5.24). In both equations, $\alpha > 0$ is an arbitrary control gain.

**Proposition 1.** *Consider that locally, the model (5.4) closely approximates the interaction matrix $\boldsymbol{L}_i = \hat{\boldsymbol{L}}_i$. For $M$ number of linearly independent displacement vectors $\delta \boldsymbol{r}$ such that the interaction matrix $\hat{\boldsymbol{L}}_i$ is invertible, the update rule (5.25) asymptotically minimizes the error $\boldsymbol{e}_i = \boldsymbol{s}^* - \boldsymbol{s}_i$.*

*Proof.* With $\delta \boldsymbol{s}_i = \boldsymbol{s}_{i+1} - \boldsymbol{s}_i$, we can write (5.4) in discretized form as

$$\boldsymbol{s}_{i+1} = \boldsymbol{s}_i + \boldsymbol{L}_i \delta \boldsymbol{r}_i. \tag{5.27}$$

The error dynamic

$$\boldsymbol{e}_{i+1} = (1 - \alpha) \boldsymbol{e}_i \tag{5.28}$$

is asymptotically stable for $\alpha \in ]0; 1[$. This can be proved by considering the Lyapunov function

$$\mathcal{V}(\boldsymbol{e}) = \boldsymbol{e}^T \boldsymbol{e}. \tag{5.29}$$

Using the error dynamic (5.28), one can derive:

$$\begin{aligned}\Delta\mathcal{V} &= \mathcal{V}(\boldsymbol{e}_{i+1}) - \mathcal{V}(\boldsymbol{e}_i) \\ &= \boldsymbol{e}_i^T((1-\alpha)^2 - 1)\boldsymbol{e}_i < 0.\end{aligned} \tag{5.30}$$

This proves the asymptotic stability of the error $\boldsymbol{e}$ locally by our inputs. ■

### 5.3.6 Model Adaptation

Since both the projection matrix $\boldsymbol{U}^T(n)$ and the interaction matrix are local approximations of the full nonlinear mapping, they need to be updated constantly. We choose a receding window approach with the window size equal to $M$.

At current instance $i$, we estimate the projection matrix $\boldsymbol{U}_i^T$ and local interaction matrix $\boldsymbol{L}_i$ with $M$ samples of the most recent data. Using the interaction matrix, and the a local target $\boldsymbol{c}_i^*$, we can derive the one-step command $\delta\boldsymbol{r}_i$ by (5.25). Once we execute the motion $\delta\boldsymbol{r}_i$, a new contour data $\boldsymbol{c}_{i+1}$ is obtained. We move to the next instance $i+1$. A new pair of input and shape data $[\delta\boldsymbol{r}_i, \boldsymbol{c}_{i+1}]$ is obtained. We shift the window by deleting the oldest data in the window and add in the new data pair. Then, using the shifted window, we compute one step control at $i+1$ instance.

The receding window approach ensures that, at each instance, we are using the latest data to estimate the interaction matrix. The overall algorithm is initialized with small random motions around the initial configuration. First, $M$ samples of shape data and the corresponding robot motions are collected. With this initialization, we can simultaneously solve for the projection matrix and estimate the initial interaction matrix using the methods described in Sect. 5.3.1 and 5.3.4. Using the projection matrix and the initial/desired shapes, we can then find an intermediate target as explained in Sect. 5.3.3.

We consider quasi-static deformation. Hence, at each iteration the system is in equilibrium and can be linearized according to (5.4). The data that best captures the current system are the most recent ones. The choice of $M$ is a trade-off between locality and richness. For fast varying deformations[1], we would expect to reduce $M$ as larger $M$ will hinder the locality assumption. Yet, if $M$ is too small, it affects the estimation of $\hat{\boldsymbol{L}}_i$ (refer to detailed discussion in Sect. 5.3.4).

## 5.4 Simulation results

In this section, we present the numerical simulations that we ran to validate our method.

[1]The notion of fast or slow varying depends on both the speed of manipulation, and on the objects deformation characteristics (which affect the rate of change in shapes) with regard to the image processing time.

### 5.4.1 Simulating the Objects

We ran simulations on MATLAB (R2018b) with two types of objects: a rigid box and a deformable cable, both constrained to move on a plane. The rigid object is represented by a uniformly sampled rectangular contour. The controllable inputs are its position and orientation. For the cable, we developed a simulator, which is publicly available at https://github.com/Jihong-Zhu/cableModelling2D. The simulator relies on the differential geometry cable model introduced in [Wakamatsu and Hirai, 2004], with the shape defined by solving a constrained optimization problem. The underlying principle is that the object's potential energy is minimal for the object's static shape [Wakamatsu et al., 1995]. Position and orientation constraints – imposed at the cable ends – are input to the simulator. The output is the sampled cable. Figures 5.4 - 5.6, 5.9, 5.10 show simulated shapes of cables and rigid boxes. We choose $K = 50$ samples for both rigid objects and cables. The camera perspective projection is simulated, with optical axis perpendicular to the plane.

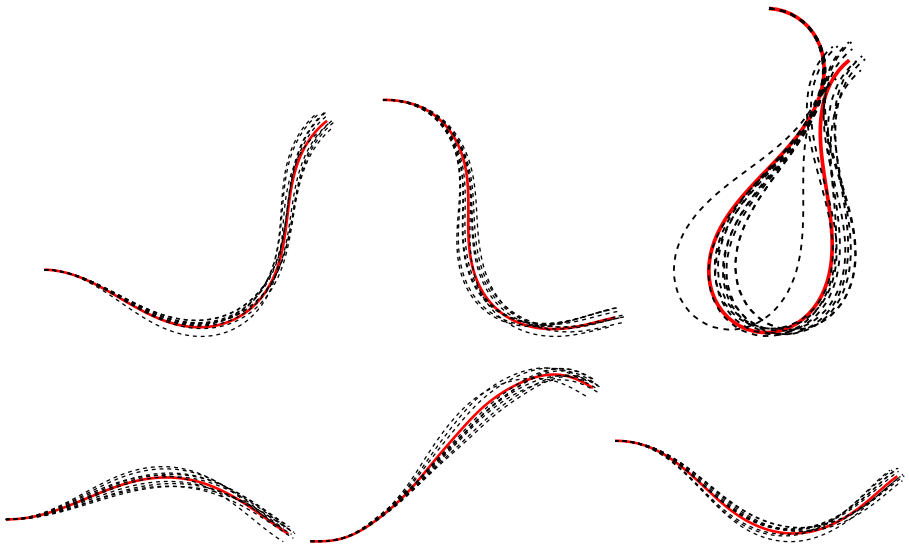


Figure 5.4: Six trials conducted to test various choices of feature dimension $k$ for a cable. In each sub-figure, the solid red lines are the initial shapes and the dashed black are the shapes resulting from 10 random motions of the right tip (translations limited to $\pm 5\%$ of the length, rotations limited to $\pm 5°$).

### 5.4.2 Selecting the Feature Dimension $k$

To check condition $k = n$ (see Sect. 5.3.2), we simulate 6 trials with distinct initial shapes of a cable. The dimension of the robot motion vector $\delta \boldsymbol{r}$ is $n = 3$ (two translations and one rotation of the right tip), and the motions are limited (each translation to $\pm 5\%$ of the cable length and the rotation to $\pm 5°$). For each trial, we command $M = 10$ random

motions around the initial shape using our simulator. Figure 5.4 shows the 6 initial cable shapes (solid red) and resulting shapes from 10 random movements (dashed black).

For each trial, we apply PCA to map the cable contour $\boldsymbol{c} \in \mathbb{R}^{2K}$ to feature vector $\boldsymbol{s} \in \mathbb{R}^k$, as explained in Sect. 5.3.1. We do this for $k = 1$, 2 and 3 and for each of these 18 experiments, we calculate the explained variance $\Upsilon(k)$ with (5.9). Table 5.1 shows these explained variances. In all 6 trials, $k = n = 3$ yields explained variances very close to 1. This result confirms that choosing $k = n$ as the dimension of the feature vector gives an excellent representation of the shape data. It is also possible to select $k = 2$, since the first two components can represent more than 99% of the variance. Nevertheless, the simulation is noise-free. Therefore, although $\Upsilon(k)$ increases little from $k = 2$ to $k = 3$, this increase is not related to noise but to an actual gain in data information.

Table 5.1: Explained variance $\Upsilon(k)$ for the 6 trials with small motion.

| | trial 1 | trial 2 | trial 3 | trial 4 | trial 5 | trial 6 |
|---|---|---|---|---|---|---|
| $k = 1$ | 0.727 | 0.795 | 0.871 | 0.847 | 0.847 | 0.705 |
| $k = 2$ | 0.992 | 0.995 | 0.996 | 0.997 | 0.997 | 0.994 |
| $k = 3$ | 0.999 | 0.999 | 0.999 | 0.999 | 0.999 | 0.999 |

At this stage, it is legitimate to ask how does this scale to *larger* movements? Fig. 5.5 illustrates 10 cable shapes generated by large movements (angle variation: $[-\frac{\pi}{2}, \frac{\pi}{2}]$, maximum translation: 106%). Again, we apply PCA ($M = 10$); Table 5.2 shows the $\Upsilon(k)$ resulting from various values of $k$.

Table 5.2: Explained variance $\Upsilon(k)$ computed with large motion.

| k | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $\Upsilon(k)$ | 0 | 0.5444 | 0.7218 | 0.8927 | 0.9919 | 0.9990 |

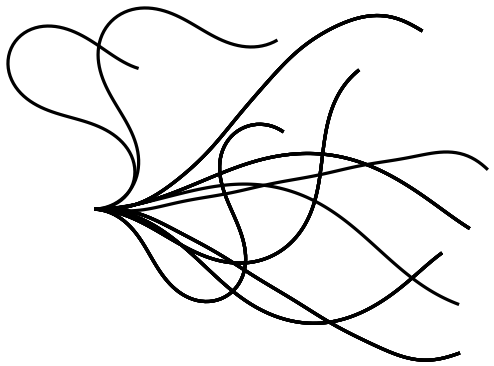


Figure 5.5: Ten distinctive cable shapes generated by large motion: angle variation: $[-\frac{\pi}{2}, \frac{\pi}{2}]$, maximum translation: 106% of the cable length.

Comparing Tables 5.1 and 5.2, it is noteworthy that $\Upsilon(4)$ with large motion is smaller than $\Upsilon(2)$ with small motion. There are two possible explanation here. One is that when

shapes stays local, the local linear mapping $\boldsymbol{L}$ in (5.4) remains constant and we need less features to characterize it; the more the shape varies, the more more features we need. Another possible explanation is that for larger motions, $M = 10$ shapes may be insufficient for PCA. Likely, the larger the changes, the larger the number of shapes $M$ needed for proper PCA.

### 5.4.3 Manipulation of Deformable Objects

With our cable simulator, we can now test the controller to modify the shape from an initial to a desired one. Again, the left tip of the cable is fixed, and we control the right tip with $n = 3$ degrees of freedom (two translations and one rotation). Using the methods described in Sect. 5.3, we choose window size $M = 5$, the Tikhonov factor $\lambda = 0$, the local target threshold $\eta = 0.8$, the control gain $\alpha = 0.01$. To quantify the effectiveness of our algorithms in driving the contour to $\boldsymbol{c}^*$, we define a scalar measure: the Average Sample Error (ASE). At iteration $i$, with current contour $\boldsymbol{c}_i$ it is:

$$\text{ASE} = \frac{\|\boldsymbol{c}_i - \boldsymbol{c}^*\|_2}{2K}. \tag{5.31}$$

A small ASE indicates that the current contour is near the desired one. In Sect. 5.3.5, we have proved that our controller asymptotically stabilizes the feature vector, $\boldsymbol{s}$ to $\boldsymbol{s}^*$. Hence, since we have also shown that $\boldsymbol{s}$ is a "very good" representation" of $\boldsymbol{c}$, we also expect our controller to drive $\boldsymbol{c}$ to $\boldsymbol{c}^*$, thus ASE to 0. This measure is also used in the real experiments.

Using the cable simulator, we compared the convergence of two control laws we proposed (5.25) and (5.26) against a baseline algorithm in [Zhu et al., 2018] which uses Fourier parameters as feature. To make methods compatible, we choose first order Fourier approximation. Note that this results in a feature vector of dimension of 6 (see [Zhu et al., 2018]) which is still twice the number $k$ used in our method. We also normalized the computed control action and then multiplied with the same gain factor 0.01.

Figure 5.6 shows the evolution of the cable shapes towards the target using (5.26). Figure 5.7 compares convergence of our methods against the Fourier-based method. We can observe that our method provides compatible convergence using half the features. Also directly computing of the inverse (5.26) provides faster convergence than (5.25). It is noteworthy to point out that the Fourier-based method requires a different parameterization for closed and open contours (see [Navarro-Alarcon and Liu, 2017] and [Zhu et al., 2018]), where in our framework, the parameterization can be kept the same.

Taking a step further, we consider the Broyden update law [Broyden, 1965], which has been used to update the interaction matrix in classic visual servoing [Hosoda and Asada, 1994, Jagersand et al., 1997, Chaumette and Hutchinson, 2007b] and shape servoing [Navarro-Alarcon et al., 2013a]. Let us hereby show why it is not applicable in our

framework.

The Broyden update is an iterative method for estimating $\boldsymbol{L}_i$ at iteration $i$. Its standard discrete-time formulation is:

$$\hat{\boldsymbol{L}}_i = \hat{\boldsymbol{L}}_{i-1} + \beta \frac{\delta \boldsymbol{s}_{i-1} - \hat{\boldsymbol{L}}_i \delta \boldsymbol{r}_{i-1}}{\delta \boldsymbol{r}_{i-1}^T \delta \boldsymbol{r}_{i-1}} \delta \boldsymbol{r}_{i-1}^T, \ \forall \boldsymbol{r}_{i-1} \neq \boldsymbol{0} \tag{5.32}$$

with $\beta \in [0; 1]$ an adjustable gain. Using our simulator, we estimate the interaction matrix using both Broyden update (with three different values of $\beta$) and our receding horizon method (5.22). We then compare (with $\hat{\boldsymbol{L}}$ estimated with either method) the one-step prediction of the resulting feature vector:

$$\hat{\boldsymbol{s}}_{i+1} = \hat{\boldsymbol{L}}_i \boldsymbol{r}_i + \boldsymbol{s}_i, \tag{5.33}$$

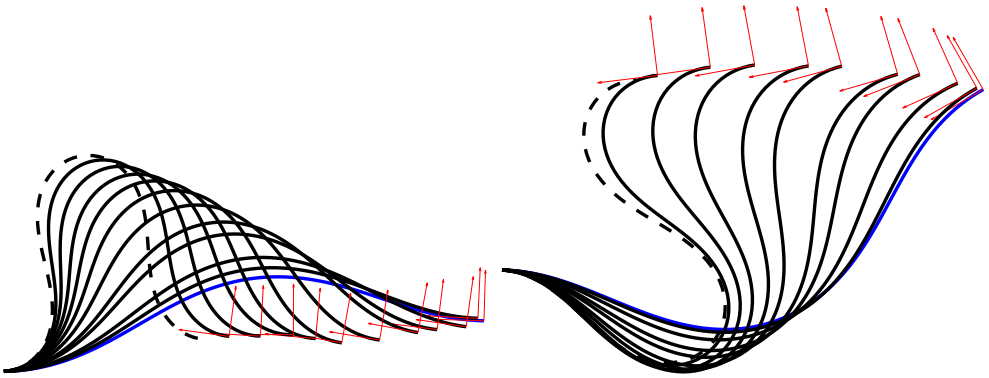


Figure 5.6: Cable manipulation with a single end-effector, moving the right tip. The blue and black lines are the initial and intermediate shapes, respectively, and the dashed black line is the target shape. The red frame indicates the end-effector position and orientation generated by our controller.

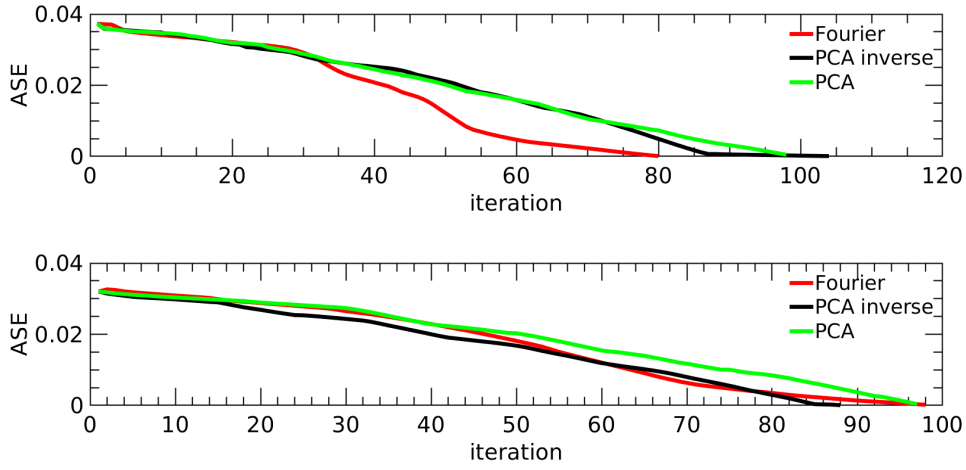
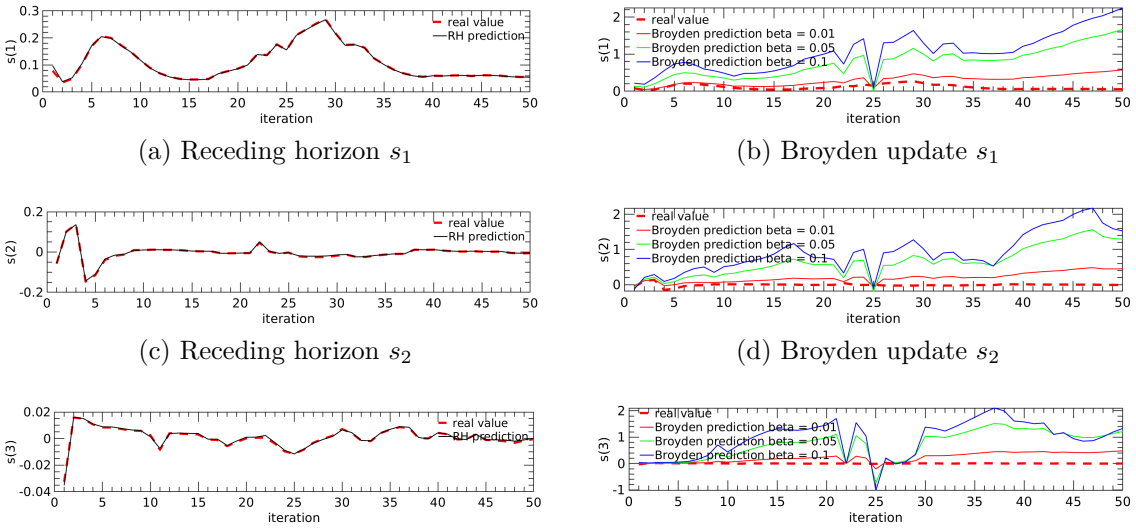



Figure 5.7: The evolution of ASE of the simulated cable manipulation using our method against the Fourier-based method as baseline. Top: left simulation in Fig. 5.6. Bottom: right simulation in Fig. 5.6.

with the ground truth $\boldsymbol{s}_{i+1}$ from the simulator. The results (plotted in Fig. 5.8) show that receding horizon outperforms all three Broyden trials. One possible reason is that the components of $\boldsymbol{s}$ fluctuate since (at each iteration) a new matrix $\boldsymbol{U}$ is used. These variations cause the Broyden method to accumulate the result from old interaction matrices, and therefore perform badly on a long term. This result contrasts with that of [Navarro-Alarcon et al., 2013a], where the Broyden method performs well since there is a fixed mapping from contour data to feature vector. Another advantage of the receding horizon approach is that it does not require any gain tuning.




(a) Receding horizon $s_1$

(b) Broyden update $s_1$

(c) Receding horizon $s_2$

(d) Broyden update $s_2$

(e) Receding horizon $s_3$

(f) Broyden update $s_3$

Figure 5.8: Comparison – for estimating $\boldsymbol{s}$ – of the receding horizon approach (RH, left) and of the Broyden update (right, with three values of $\beta$). The topmost, middle and bottom plots show the one step prediction of $s_1$, $s_2$ and $s_3$, respectively. In all plots, the dashed red curve is the ground truth from the simulator. The plots clearly show that the receding horizon approach outperforms all three Broyden trials.

### 5.4.4 Manipulation of Rigid Objects

The same framework can also be applied to rigid object manipulation. Consider the problem of moving a rigid object to a certain position and orientation via visual feedback. This time, the shape of the object does not change, but its pose will (it can translate and rotate). We use the same $M$, $\lambda$, $\eta$ and $\alpha$ as for cable manipulation. We compare the convergence of two control laws proposed (5.25) and (5.26) against a baseline using image moments [Chaumette, 2004]. The translation and orientation can be represented with image moments and the analytic interaction matrix can be computed as explained in [Chaumette, 2004]). To make methods compatible, we normalize the computed control and then multiply it by the same factor 0.01.

Figure 5.9 shows two simulations where our controller successfully moves a rigid object from an initial (blue) to a desired (dashed black) pose using control law (5.26). Figure

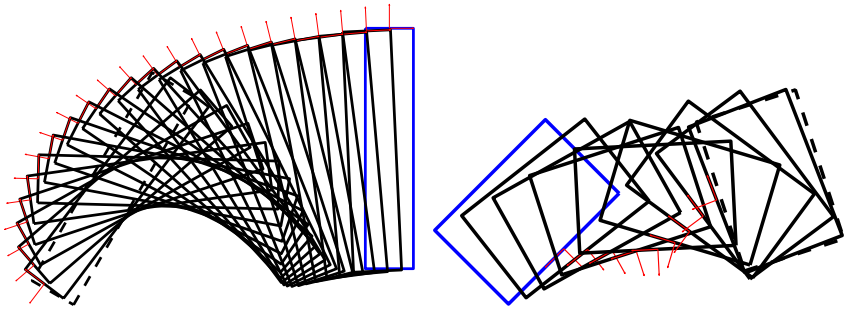


Figure 5.9: Manipulation of a rigid object with a single end-effector (red frame). The initial, intermediate and desired contours are respectively blue, solid black and dashed black. Note that in both cases, our controller moves the object to the desired pose.

5.11 compares convergence of our methods against the image moments method. We can observe that our method provides a slightly slower convergence. Directly computing the inverse (5.26) provides a convergence similar to (5.25). Later, we will show why our method is slower. Yet, the fact that it can be applied on both deformable and rigid objects makes it stand out over the other techniques.

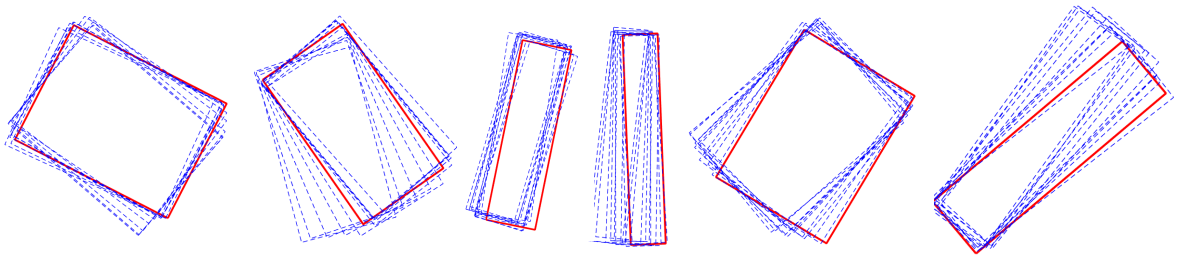


Figure 5.10: From an initial (red) pose, we generate 10 (dashed blue) random motions of a rigid object.

Taking a step further, we would like to analyze locally what does each component of the feature vector represent. To this end, we apply $M = 10$ random movements (rotation range $[-0.11, 0.09]$, maximum translation 15% of the width) to a rigid rectangular object of length 5 times larger than width (see Fig. 5.10). We compute the projection matrix as explained in Sect. 5.3.1, and transform the contour samples to feature vectors. Following the rationale explained in Sect. 5.3.2, we set $k = n = 3$. Then, we seek the relationship – at each iteration – between the object pose $x$, $y$, $\theta$ and the components of the feature vector $\boldsymbol{s}$ generated by PCA. To this end, we use bivariate correlation [Feller, 2008] defined by:

$$\rho = \frac{E[(\xi - \bar{\xi})(\zeta - \bar{\zeta})]}{\sigma_\xi \sigma_\zeta}, \tag{5.34}$$

where $\xi$ and $\zeta$ are two variables with expected values $\bar{\xi}$ and $\bar{\zeta}$ and standard deviations $\sigma_\xi$ and $\sigma_\zeta$. An absolute correlation $|\rho|$ close to 1 indicates that the variables are highly cor-

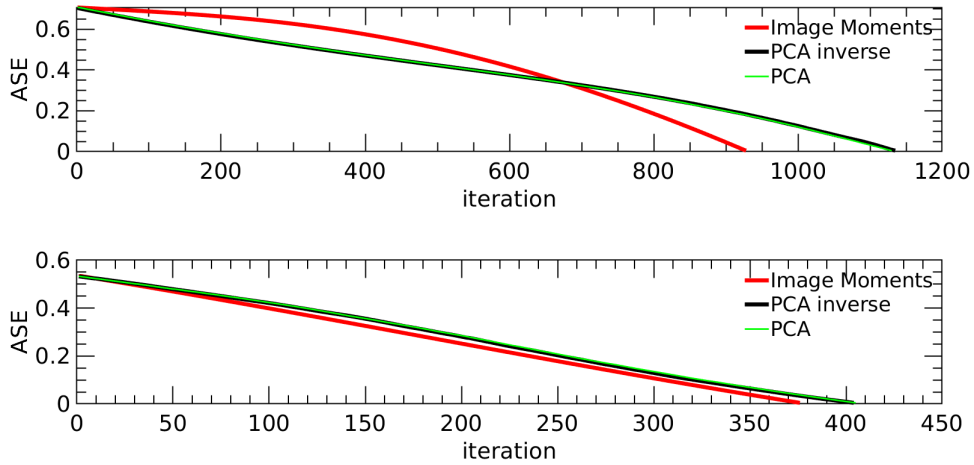



Figure 5.11: Evolution of ASE of the simulated rigid object manipulation using our method against image moments. Top: left simulation in Fig. 5.9, Bottom: right simulation in Fig. 5.9.

related. All the simulations in Fig. 5.10 exhibit similar correlation between the computed feature vector and the object pose. In Table 5.3, we show one instance (Left first simulation in Fig. 5.10) of the correlation between variables, with high absolute correlations marked in red. It is clear from the table that each component in the feature vector relates strongly to one pose parameter. We further demonstrate the correlation in Fig. 5.12, where we plot the evolution of object poses and feature components. Note that $s_2$ and $\theta$ are negatively correlated. The slower convergence (compared to image moments) could be as a result of non-unitary correlation between extracted features and object pose.

Table 5.3: Correlation $\rho$ between $s_1$, $s_2$, $s_3$ and $x$, $y$, $\theta$.

| | $x$ | $y$ | $\theta$ |
|---|---|---|---|
| $s_1$ | -0.2819 | -0.3343 | 0.9887 |
| $s_2$ | 0.2607 | -0.8547 | -0.0465 |
| $s_3$ | 0.9230 | 0.3629 | -0.1426 |

## 5.5 Experiments

Figure 5.13 outlines our experimental setup. We use a KUKA LWR IV arm. We constrain it to planar ($n = 3$) motions $\delta\boldsymbol{r}$, defined in its base frame (red in the figure): two translations $\delta x$ and $\delta y$ and one counterclockwise rotation $\delta\theta$ around $z$. A Microsoft Kinect V2 observes the object[2]. A Linux-based 64-bit PC processes the image at 30 fps. In the following sections, we first introduce the image processing for contour extraction, then present the experiments.

[2]We only use the RGB image – not the depth – from the sensor.

### 5.5.1 Image Processing

This section explains how we extract and sample the object contours from an image. We have developed two pipelines, according to the kind of contours (See Fig. 5.14): open (e.g., representing a cable) and closed. We hereby describe the two.

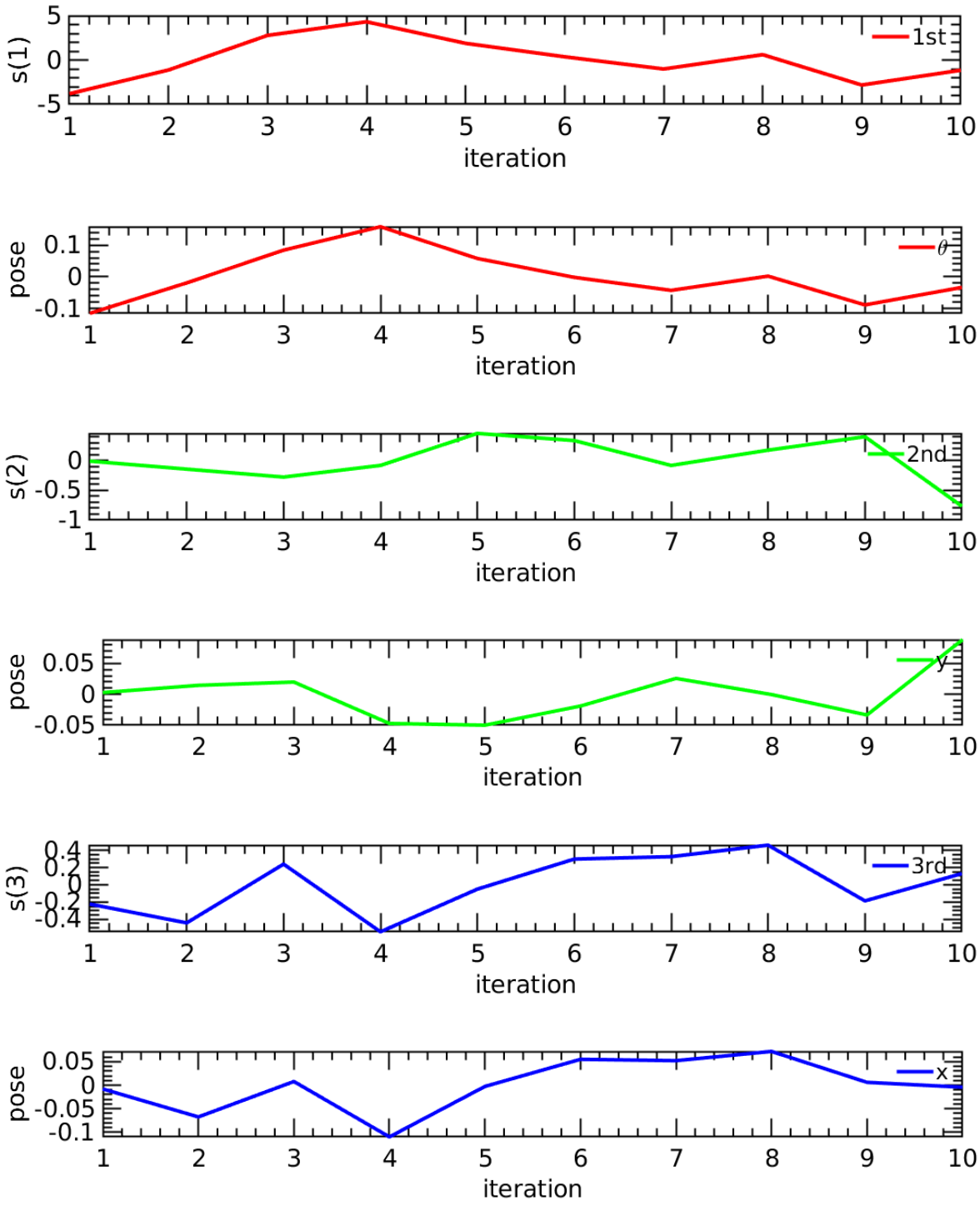



Figure 5.12: Progression of the auto-generated feature components (row 1, 3, 5: $s_1$, $s_2$, $s_3$) vs. object pose (row 2, 4, 6: $x$, $y$, $\theta$). We have purposely arranged the variables with high correlation with the same color.

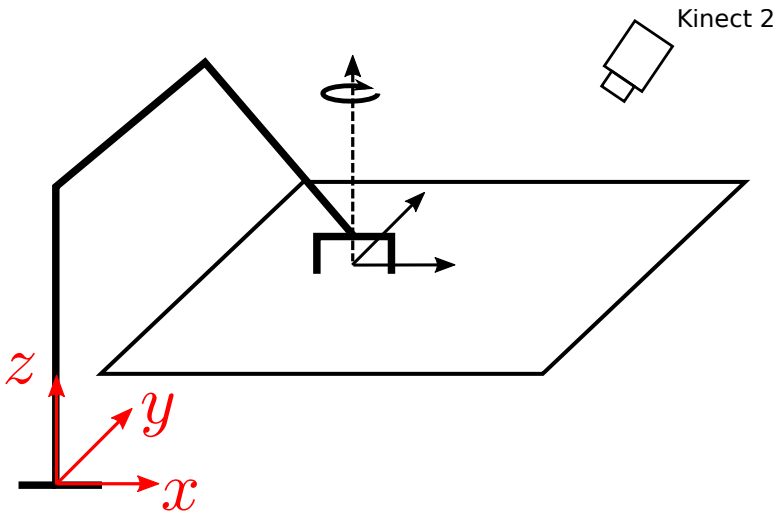



Figure 5.13: Overview of the experimental setup.

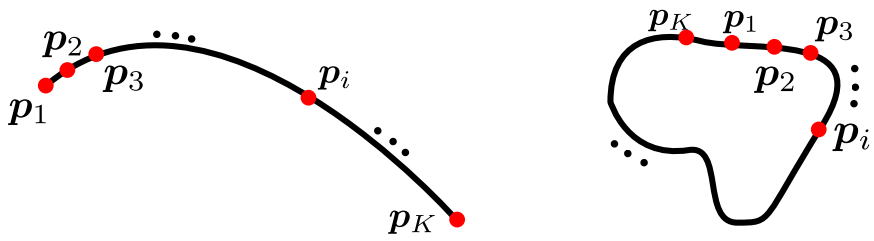
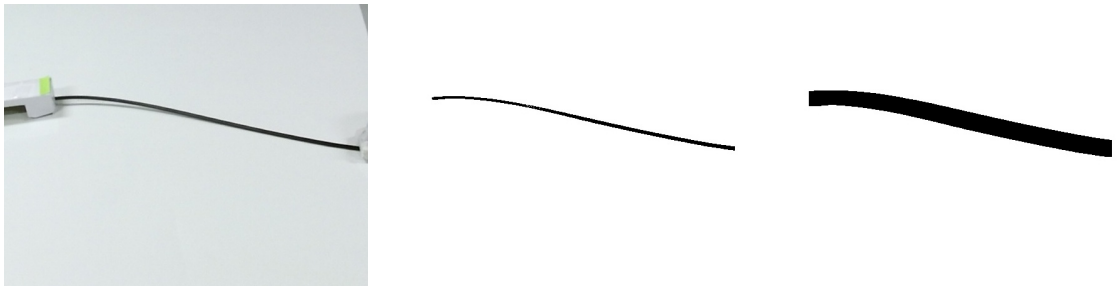


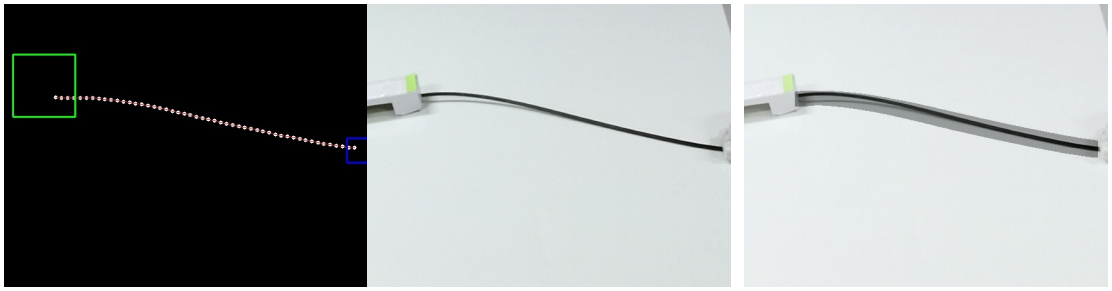

Figure 5.14: Open (left) and closed (right) contours can be both represented by a sequence of sample pixels in the image.

#### 5.5.1.1 Open Contours

The overall pipeline for extracting an open contour is illustrated in Fig. 5.15 and **Algorithm 2**. On the initial image, the user manually selects a Region of Interest (ROI, see Fig. 5.15a) containing the object. We then apply thresholding, followed by a morphological opening, to the image, to remove the noise and obtain a binary image as in Fig. 5.15b. This image is dilated to generate a mask (Fig. 5.15c), which is used as the new ROI to detect the object on the following image. Figure 5.15e is the object after a small manipulation motion and 5.15f shows the mask with the grey color which contains the cable in Fig. 5.15e. On each binary image, we apply **Algorithm 2** to uniformly sample the object (see Fig. 5.15d, where the green box indicates the end-effector). This is the contour $\boldsymbol{c}$ used by our controller.



(a) ROI (b) After thresholding (c) Mask

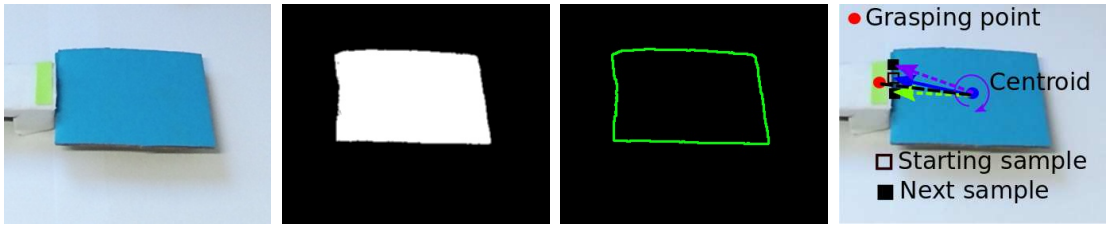


(d) Sampled data (e) Next image (f) Cable in the mask

Figure 5.15: Image processing steps needed to obtained the sampled open contour of an object (here, a cable).

#### 5.5.1.2 Closed Contours

The procedure is shown in Fig. 5.16. For an object with uniform color (in the experiment blue), we apply HSV segmentation, followed by Gaussian blur of size 3, and finally the OpenCV *findContour* function, to get the object contour. The contour is then re-sampled using **Algorithm 2**. The starting point and the order of the samples is determined by tracking the grasping point (red dot in Fig. 5.16d) and the centroid of the object (blue dot). We obtain the vector connecting the grasping point to the centroid. Then, the starting sample is the one closest to this vector, and we proceed along the contour




(a) (b) (c) (d)

Figure 5.16: Image processing for getting a sampled closed contour: (a) original image, (b) image after thresholding and Gaussian blur, (c) extracted contour, (d) finding the starting sample and the order of the samples.

clockwise.

**Algorithm 2** Generate fixed number of sampled data
*Input*: $\boldsymbol{P}' = [\boldsymbol{p}'_1, \cdots, \boldsymbol{p}'_L]$, original ordered sampled data and $K$, desired number of data samples generated
*Output*: $\boldsymbol{P} = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_K]$, re-sampled data.

compute the full length of $\mathcal{L}$ of $\boldsymbol{P}'$.
compute desired distance per sample: $\mu = \mathcal{L}/K$
$k = 1,\ h = 1$
dist $= 0$
$\boldsymbol{p}_k = \boldsymbol{p}'_l$
**while** $k \leq K$ **do**
    $\boldsymbol{p}_{\text{next}} = \boldsymbol{p}'_{h+1}$
    $d = ||\boldsymbol{p}'_{k+1} - \boldsymbol{p}_{\text{next}}||_2$
    **if** $d + \text{dist} \leq \mu$ **then**
        dist $=$ dist $+ d$
        $\boldsymbol{p}_{k+1} = \boldsymbol{p}_{\text{next}}$
        $k = k + 1,\ h = h + 1$
    **else**
        $\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \mu \frac{\boldsymbol{p}'_{k+1} - \boldsymbol{p}_{\text{next}}}{d}$
        $k = k + 1$
        dist $= 0$

### 5.5.2 Vision-based Manipulation

In this section, we present the experiments that we ran to validate our algorithms, also visible at https://youtu.be/gYfO2ZxZ5KQ. To demonstrate the generality of our framework, we tested it with:

- Rigid objects represented by closed contours,
- Deformable objects represented by open contours (cables),
- Deformable objects represented by closed contours (sponges).

We carried out different experiments with a variety of initial and desired contours and camera-to-object relative poses. The variety of both geometric and physical properties demonstrates the robustness of our framework. The variety of camera-to-object relative poses shows that—as usual in image-based visual servoing [Chaumette and Hutchinson, 2006b]—camera calibration is unnecessary. The algorithm and parameters are the same in all experiments; the only differences are in the image processing, depending on the type of contour (closed or open, see Sect. 5.5.1).

We obtain the desired contours by commanding the robot with predefined motions. Once the desired contour is acquired, the robot goes back to the initial position, and then should autonomously reproduce the desired contour. Again, we set the number of features $k = n = 3$, and use $K = 50$ samples to represent the contour $\boldsymbol{c}$. We set the window size $M = 5$, both for obtaining the feature vector $\boldsymbol{s}$ and the interaction matrix $\boldsymbol{L}$. The control gain is 0.01, the local target threshold $\eta = 0.8$ and the Tikhonov factor $\lambda = 0.01$. At the beginning of each experiment, the robot executes 5 steps of small[3] random motions to obtain the initial features and interaction matrix.

For all the experiments, we set the same termination condition at iteration $i+1$ using ASE defined in (5.31) such that:

1. $\text{ASE}_i < 1$ pixel and
2. $\text{ASE}_{i+1} \geq \text{ASE}_i$.

In the graphs that follow, we show the evolution of ASE in blue before the termination condition, and in red after the condition (until manual stop by the operator).

Figure 5.17 presents 8 experiments, one per column. Columns 1 – 3, 4 – 6 and 7 – 8 show respectively manipulation of: cable, rigid object and sponge. The first row presents the full RGB image obtained from Kinect V2. The second and third rows zoom in on the manipulation at the initial and final iterations. We track the end-effector in the image with a green marker for contour sampling. The desired and current contours are drawn in red and blue, respectively.

Figure 5.18 shows the decreasing trend of error ASE for each experiment. The initial increase of ASE in the experiments can be due to the random motion at the beginning of the experiments. In general, we found that ASE is more noisy for the closed than for the open contour. This discontinuity is visible in Figures 5.18c and 5.18d (zigzag evolution). Such noise is likely introduced by the way we sampled the contour. When we have false contour data, the value of ASE may encounter a sudden discontinuity. Figure 5.19 shows examples of these false samples, output by the image processing pipeline. Despite these errors, thanks to the "forgetting nature" of the receding horizon and to the relatively small window size ($M = 5$), the corrupted data will soon be forgotten, and it will not

[3]The notion of small is relative, and usually dependent on the size of the object the robot is manipulating. Refer to Sect. 5.3.1 (especially Fig. 5.4) for a discussion on this.

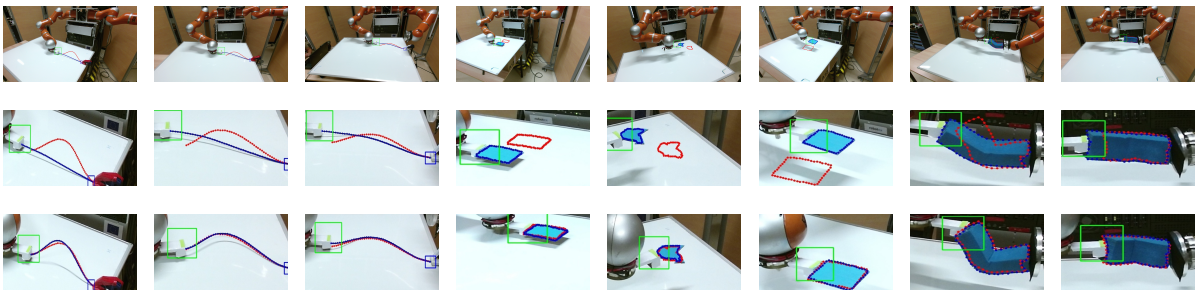
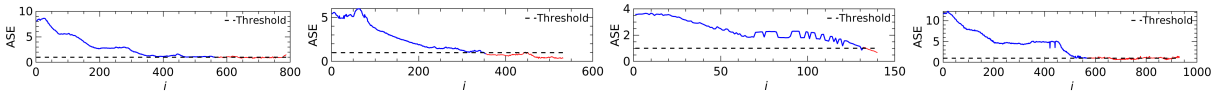
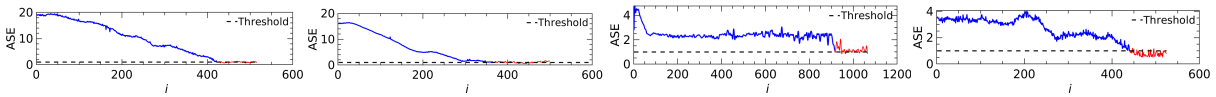


Figure 5.17: Eight experiments with the robot manipulating different objects. From left to right: a cable (columns 1 – 3), a rigid object (columns 4 – 6) and a sponge (columns 7 and 8). The first row shows the full Kinect V2 view, and the second and the third columns zoom in to show the manipulation process at the first and last iterations. The red contour is the desired one, whereas the blue contour is the current one. The green square indicates the end-effector.




(a) cable – column 1 (b) cable – column 2 (c) cable – column 3 (d) rigid object – column 4




(e) rigid object – column 5 (f) sponge – column 6 (g) sponge – column 7 (h) sponge – column 8

Figure 5.18: Evolution of $e_i$ at each iteration $i$, for the 8 experiments of Fig. 5.17. The black dashed lines indicate the threshold ASE = 1 pixel. The blue curves show $e_i$ until the termination condition, whereas the red curves show the error until manual termination by the human operator.

hinder the overall manipulation task. Yet, the overall framework would benefit from a more robust sensing strategy, as in [Chi and Berenson, ].

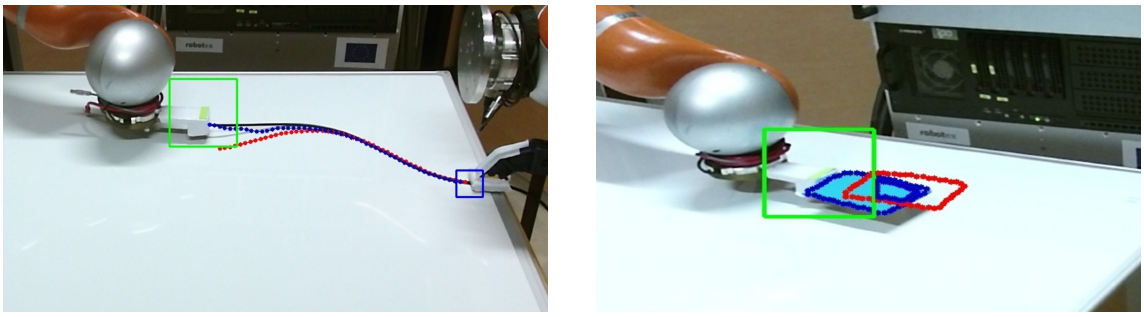
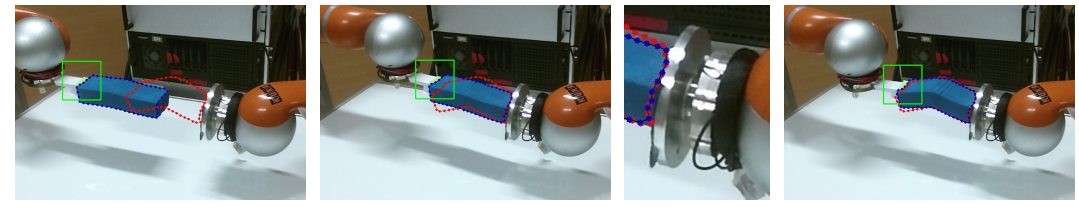
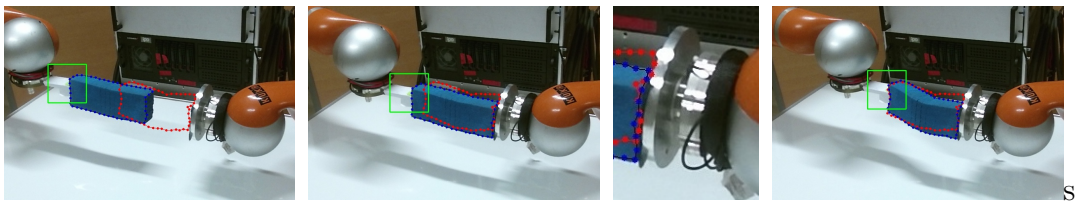


Figure 5.19: False contour data from the image can cause noise in ASE.



(a) (b) (c) (d)

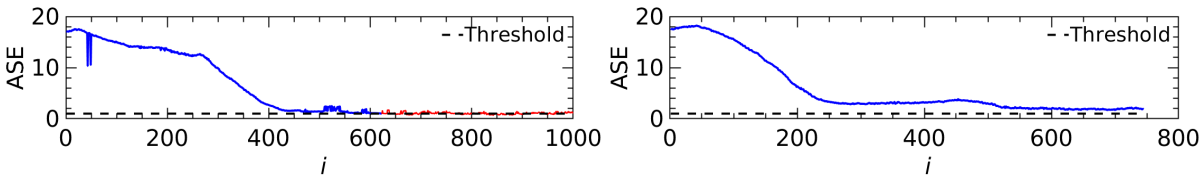


(e) (f) (g) (h)

Figure 5.20: Two "move and shape" experiments grouped into two rows. The desired contour (red dotted) is far from the initial one. This requires the robot to 1) move the object, establish contact with the right – fixed – robot arm, 2) give the object the desired shape, by relying on the contact. The first column shows the starting configuration, the second column presents the contact establishment, and the third column zooms in to show the alignment. The last column shows the final results.

Finally, since our framework can deal with both rigid and deformable objects, we tested it in two experiments where the same object (a sponge) can be both rigid (in the free space), and deformed (when in contact with the environment). These experiments require the robot to: 1) move the object, establish contact, 2) give the object the desired shape, by relying on the contact. Figure 5.20 presents these two original "move and shape" servoing experiments with the corresponding errors ASE plotted in Fig. 5.21. We use a second fixed robot arm to generate the deforming contact. As the figures and curves show, both experiments were successful.

The success of the "move and shape" task is largely dependent on the contact establishment. However, even when the initial contact has some misalignment (see Fig. 5.20c




(a) move and shape – row 1

(b) move and shape – row 2

Figure 5.21: The evolution of $e_i$ for the experiments of Fig. 5.20. The black dashed line indicates the threshold ASE = 1. The blue curves show $e_i$ until the termination condition, whereas the red curves show the error until manual termination by the human operator.

and 5.20g), our framework can still reduce the ASE to give a reasonable final configuration (see Fig. 5.20h and Fig. 5.21b).

## 5.6 Conclusion

In this chapter, we propose algorithms to automatically and concurrently generate object representations (feature vectors) and models of interaction (interaction matrices) from the same data. We use these algorithms to generate the control inputs enabling a robot to move and shape the said object, be it rigid or deformable. The scheme is validated with comprehensive experiments, including a desired contour that requires both moving and shaping. We believe it is unprecedented in previous research.

Our framework adopts a model-free approach. The system characteristics are computed online with visual and manipulation data. We do not require camera calibration, nor a priori knowledge of the camera pose, object size or shape. An open question remains the management of 6 DOFs motions of arms. Indeed, while the proposed control strategy can be easily generalized to 6 DOFs motions, it relies on a sufficiently accurate extraction of feature vectors from vision sensors. A very challenging task is to generate complete 3D feature vectors of objects from a limited sensor set, due to partial view of objects and occlusions.

The framework could benefit from robust sensing of deformation. In fact, one obvious setback is that the representation and model of interaction are extremely local. Thus, they cannot guarantee global convergence. In addition, our framework cannot infer whether a shape is reachable or not. This drawback is solvable by using a global deformation model for control. But as we mentioned earlier, a global model usually requires an offline identification phase which we want to avoid. In fact, for different objects, we will need to re-identify the model. There is a dilemma in using a global deformation model.

Maybe one of the possible solutions to this dilemma is to have both our method and deep learning based methods running in parallel. While our scheme enables fast online computation and direct manipulation, the extracted data can be used by a deep neural

network to obtain a global interaction mapping. Once a global mapping is learned, it can later be used for direct manipulation and to infer feasibility of the goal shape.

Chapter 6

# Deformable Linear Object Manipulation Planning with Contacts

Contact is an important concept in robotics. In physical Human-Robot Interaction (pHRI), contact with humans is harmful and should be avoided or reduced. Collision avoidance can be done with planning [Brock and Khatib, 2002]. In post contact phase, methods could be used for reducing the impact [De Luca et al., 2006]. Proper contact needs to be established for grasping an object [Roa and Suárez, 2015]. For humanoids, contact can be used to achieve a given posture [Kheddar et al., 2019] or execute complex tasks such as climbing stairs [Zhang et al., 2017] or a ladder [Vaillant et al., 2016]. In this chapter, we investigate the use of contacts in the context of deformable object manipulation. To the best of our knowledge, few papers to date have investigated this topic.

## 6.1 Introduction

Humans use contacts when manipulating deformable objects. For instance, when folding a towel, we will place it on a flat surface to constrain deformation due to gravity. Another example is when we make cable harnesses, usually a set of cylinder sticks are placed to regulate the path a cable will follow. Specific contacts are designed for regulating deformable objects such as hanging hooks for clothes. The reason behind the use of external contacts lies in the fact that there are infinite degrees of freedom in the object deformation [Guo et al., 2013], yet finite inputs from manipulators/hands. Often when dealing with deformable objects, humans not only apply both hands, but also contacts in the environment to regulate the object shape.

In robotic manipulation planning, contact with the environment is often considered undesired, thus should be avoided. A collision free path planner was developed in [Lamiraux and Kavraki, 2001] using a randomized algorithm. A planner in [Moll and Kavraki, 2006] computed a path in shape space from one minimal energy curve to another while satisfying environmental constraints. Bretl and McCarthy showed that

the shape space of an elastic cable is a six-dimensional smooth manifold [Bretl and McCarthy, 2014]. Later, the authors of [Borum and Bretl, 2015] took a step forward and proved path-connectedness of this space. Few papers have investigated cable manipulation with contacts in the environment. In a pioneer work [Henrich et al., 1999], the authors presented the specification of contact states for linear objects with polyhedral obstacles, and identified unstable/stable contact states. Acker and Henrich further proposed visual features for the detection of contact state transitions in cable manipulation [Acker and Henrich, 2003]. In a more recent research, the authors planned cable manipulation strategies with a simulator in both free and contact space [Rousel and Taïx, 2014]. Some works in deformable object manipulation exploited contact for manipulation: plastic material shaping [Cherubini et al., 2018] and towel folding [Maitin-Shepard et al., 2010] where some or all manipulation tasks were done with contact compensating the gravity. Yet, these papers do not study the specific use of contact.

In this chapter, we tackle a practical scenario that uses contacts for shaping DLOs. In cable harnesses, cables usually need to follow a designated path defined by a set of contacts for easy management. Figure 6.1 shows a typical cable harness board with contacts. We simplify the problem by considering circular contacts on the board. Our objective is to establish a robotic manipulation framework that automates the cable routing process. The robot plans its motion according to the contacts' placement, detects the occurrence of a contact, modifies its manipulation behaviour accordingly, and finally achieves a desired configuration of the cable.

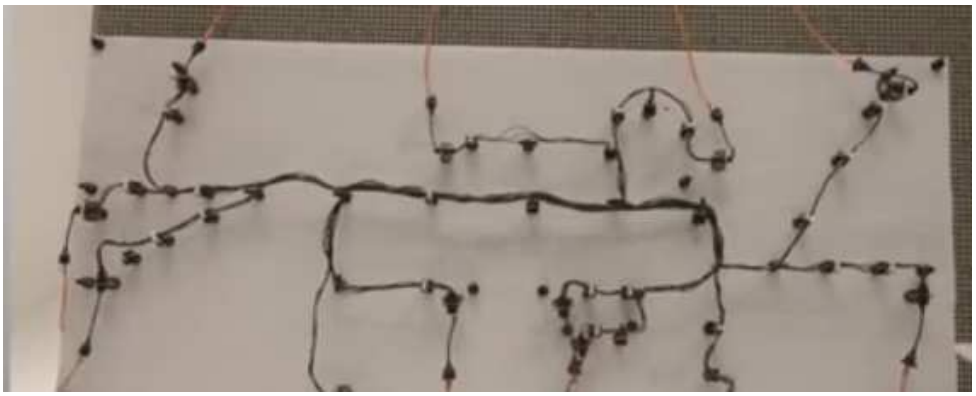


Figure 6.1: Cable routing using contact.

## 6.2 Problem Statement

The overall problem is depicted in Fig. 6.2. We consider contacts as small circles on the 2D plane. For a cable with sufficient length, given a starting end of the cable, with a list of ordered small [1] circular contacts placed on the plane, the cable should be manipulated

[1] The radius of the contact is much smaller than the length of the cable.

so that it follows the path defined by the contacts' position and sequence. The order of the contacts is known a priori by the robot. The cable must be shaped by each contact sequentially and in the end reach the target position.

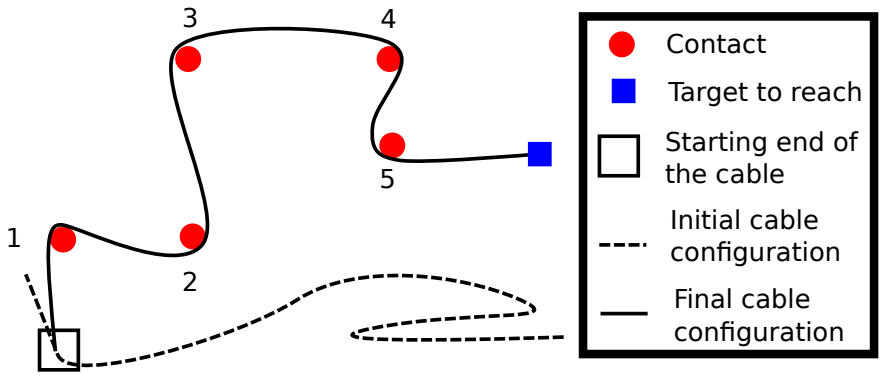



Figure 6.2: A contact-based manipulation example to illustrate the problem. The order of the contacts is given by the number besides each contact. This information is provided a priori to the robot.

The cable should touch all contacts in the given order, without creating any loop. We do not consider friction in this work, and neglect the cable's deformation along the tangential direction.

## 6.3 Our Contributions

To the best of our knowledge, contact-based cable routing relying on robotic manipulation is still an open research problem. Yet, it is widely present in the industry (e.g., in cable harness). We address this practical problem by a mathematical analysis of the contact mobility. We propose an index to quantify the mobility. This index motivates our choice of motion primitives. A motion planning framework using the primitives is then designed for the robot to shape cables by contacts. Finally, we validate experimentally the framework, and evaluate its performance using the proposed contact mobility index.

## 6.4 Framework overview

We utilize a dual arm robot for the cable manipulation task. The robot is equipped with two end-effectors $\mathcal{M}$ (mobile) and $\mathcal{F}$ (fixed). We place $\mathcal{F}$ at a fixed pose to hold the starting end of the cable. $\mathcal{M}$ holds the other end of the cable and is free to move on the 2D manipulation plane. An in-hand camera is mounted on $\mathcal{M}$ to provide visual feedback.

The framework consists of a planner and a vision-based contact detector. The planner plans the motion for $\mathcal{M}$ given contacts and a target location. The detector identifies the occurrence of a contact.

## 6.5 Angular Contact Mobility Index (ACMI)

One way of analyzing the contact is by its mobility index, which measures the free motion of the object in contact [Rimon and Burdick, 1998]. In this section, we introduce a novel contact mobility index based on the angular range of motion. We term it ACMI.

The robot manipulates the cable on a plane. The cable is soft and can adapt to the curvature of the contact object. In Fig. 6.3 we show an example. We construct a local Cartesian coordinate frame $Oxy$ at the center of the object. For any point $\boldsymbol{\rho}$ on the contact curve, the direction of the contact force is denoted $\boldsymbol{n}(\boldsymbol{\rho})$. We consider the motion of the cable relative to the contact. The direction of motion is a unit vector $\boldsymbol{v}(\phi) = [\cos\phi \ \sin\phi]^T$ with $\phi$ $(0 \leq \phi < 2\pi)$ the angle between the vector and $x$ axis.

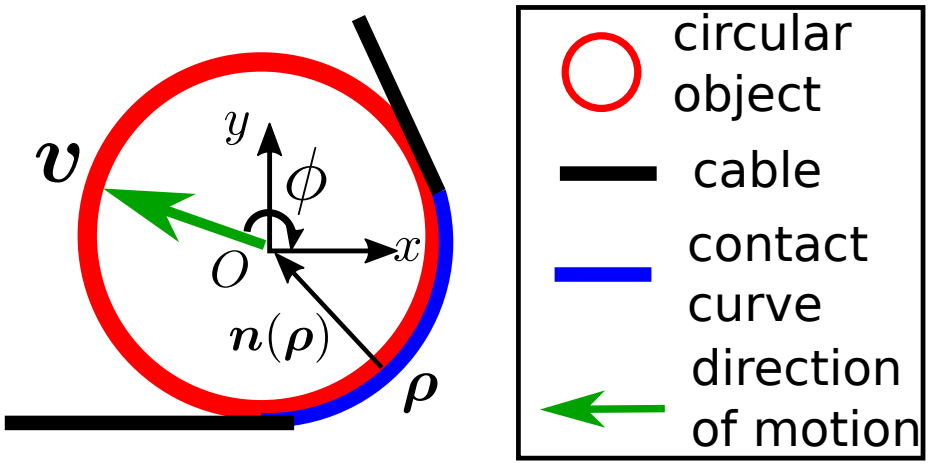



Figure 6.3: Example of a circular object in contact with a cable. The green vector is a candidate direction of relative object/cable motion. Many other directions are possible.

**Definition 1.** Angular Contact Mobility Index (ACMI): the angle (in radians) that represents the range of directions the object can move to break free from the contact.

In Fig. 6.4, we consider – in four cases – the directions of motion of the contact object (red circle) with regards to the cable. The direction $\boldsymbol{v}$ is parameterized by $\phi$. We derive the ACMI for these cases and finally provide a general expression for the ACMI.

When there is no contact (Fig. 6.4a), the break free (from contact) directions are given by the following set:

$$\mathcal{R} = \{\phi \in [0\ , 2\pi)\}\,. \tag{6.1}$$

Since the object can move in all directions, the ACMI equals $2\pi$.

For a single contact point $\boldsymbol{\rho}$ (Fig. 6.4b), the set of break free directions is defined by:

$$\mathcal{R}(\boldsymbol{\rho}) = \{\phi \in [0\ , 2\pi) : \boldsymbol{n}(\boldsymbol{\rho}) \cdot \boldsymbol{v}(\phi) > 0\}\,. \tag{6.2}$$

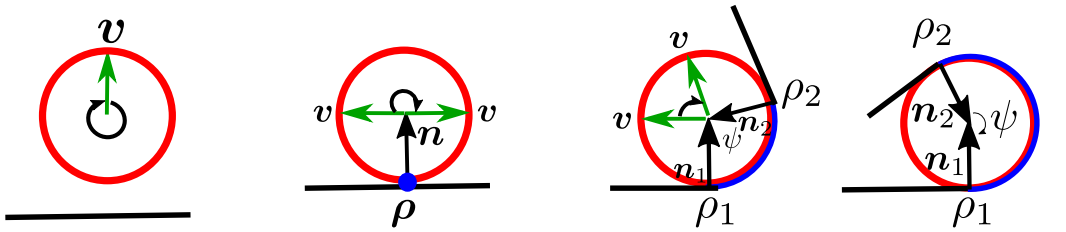
This defines an open set of directions with only positive components in the direction of $\boldsymbol{n}$. The angular range in the $\mathcal{R}(\boldsymbol{\rho})$ is $\pi$. The ACMI equals $\pi$.

For the curved contact in Fig. 6.4c, the set of break free directions is expressed by:

$$\mathcal{R} = \bigcap_{\boldsymbol{\rho}=\boldsymbol{\rho}_1}^{\boldsymbol{\rho}_2} \mathcal{R}(\boldsymbol{\rho}). \tag{6.3}$$

Noting $\psi$ the contact curve angular range, if $\psi \in (0, \pi]$ the ACMI is $\pi - \psi$.

Instead, if $\psi \in (\pi, 2\pi]$ (see Fig. 6.4d), the ACMI equals 0. No matter which direction the object moves, it cannot break free from the contact.




(a) No contact (b) Point contact (c) $0 < \psi \leq \pi$ (d) $\pi < \psi \leq 2\pi$

Figure 6.4: The ACMI in four contact cases, (a): no contact; (b): point contact; (c): curved contact with $0 < \psi \leq \pi$; (d): curved contact with $\pi < \psi \leq 2\pi$.

To sum up, we provide a quantitative definition of the ACMI:

$$\text{ACMI} = \begin{cases} 2\pi, & \text{No contact,} \\ \pi, & \text{Point contact,} \\ \max(0, \pi - \psi) & \text{Curved contact.} \end{cases} \tag{6.4}$$

Using the ACMI, we quantify the free motion range due to the contact. The higher the index, the larger the range of motion sufficient to break free from contact (i.e., it is easier to lose contact).

Below we analyze two motions of the cable and their effects.

The first motion is termed *rotation* (see Fig. 6.5). Starting from an initial contact point ($\boldsymbol{\rho}_2$ in Fig. 6.5), the ACMI is $\pi$. Consider the left segment of the cable fixed, and perform an anti-clockwise rotation with center at the contact for the right segment: we expect the contact point to become a curve. Then the length of the curve grows due to this movement, so the ACMI decreases. The reverse motion will make the contact curve shrink, and the ACMI increase.

The second motion is termed *sliding*. In case of contact point (Fig. 6.6a), sliding corresponds to:

$$\mathcal{R}(\boldsymbol{\rho}) = \{\phi \in [0\ , 2\pi) : \boldsymbol{n}(\boldsymbol{\rho}) \cdot \boldsymbol{v}(\phi) = 0\}\,. \tag{6.5}$$

For a curved contact with two end points (Fig. 6.6b), we denote two edges of the

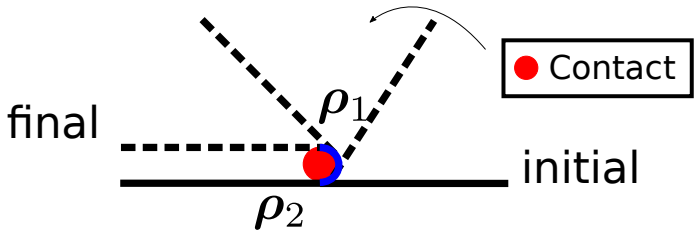



Figure 6.5: The effect of rotation on the ACMI

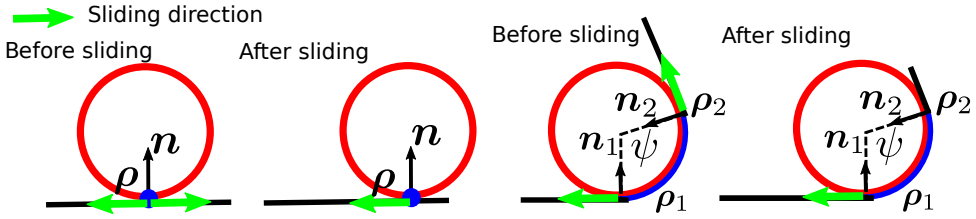
curve as $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ with $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ being the directions of the associated contact force. Sliding corresponds to the direction within:

$$\mathcal{R}_1 \bigcup \mathcal{R}_2, \tag{6.6}$$

with

$$\mathcal{R}_1 = \{\phi \in [0\ , 2\pi) : \boldsymbol{n}_1 \cdot \boldsymbol{v}(\phi) = 0\}\,,$$
$$\mathcal{R}_2 = \{\phi \in [0\ , 2\pi) : \boldsymbol{n}_2 \cdot \boldsymbol{v}(\phi) = 0\}\,.$$

In both figures we show an example of contact after sliding in a specific direction. The sliding motion maintains the contact curve (point) and the ACMI stays unchanged.




(a) Point contact case (b) Curved contact case

Figure 6.6: The effect of sliding on the ACMI.

In the next sections, we will relate robotic cable manipulation with these two motions. Using the ACMI, we show in Section 6.9.3 that with the proposed framework, the robot constructs and utilizes contacts for shaping the cable to reach a desired configuration.

## 6.6 Motion Primitives

In Fig. 6.7, we define the manipulation plane coordinate frame $\mathcal{F}xy$ with an origin at the position of the fixed end-effector $\mathcal{F}$, and we attach a local coordinate frame $\mathcal{M}xy$ to the mobile end-effector $\mathcal{M}$. The pose of $\mathcal{M}$ in $\mathcal{F}xy$ is then $\boldsymbol{q}_\mathcal{M} = [x_\mathcal{M}\ y_\mathcal{M}\ \theta_\mathcal{M}]^T \in \mathbb{R}^3$, with $\theta_\mathcal{M}$ the angle expressing orientation of $\mathcal{M}xy$ with regard to $\mathcal{F}xy$.

The fixed end-effector $\mathcal{F}$ is designed such that it can either *hold* or *release* one end of the cable. The mobile end-effector $\mathcal{M}$ always holds the other end of the cable. When $\mathcal{F}$

is holding, the length of the cable between the two end effectors is fixed. Then $\mathcal{M}$ rotates the cable (Fig. 6.7a) with a fixed point on the cable as the rotation center. When $\mathcal{F}$ releases, it allows $\mathcal{M}$ to pull the cable so that the length of the cable between the two end-effectors increases (Fig. 6.7b).

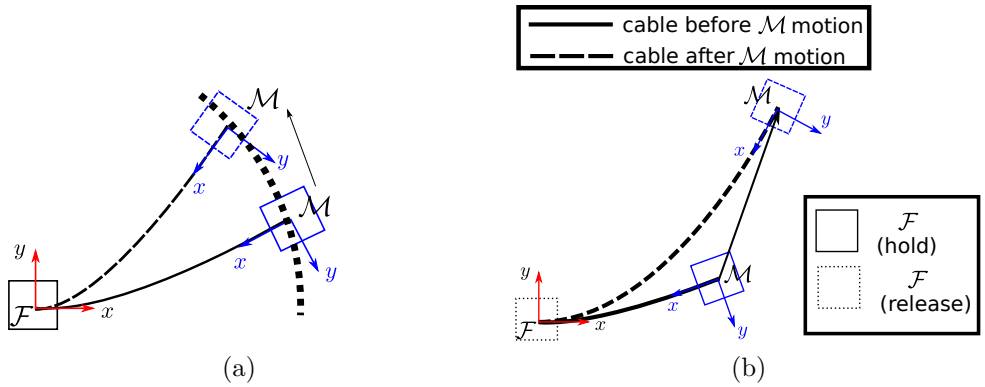



Figure 6.7: Motion primitives: (a). End-effector $\mathcal{F}$ holds while end-effector $\mathcal{M}$ rotates the cable, (b). End-effector $\mathcal{F}$ releases while end-effector $\mathcal{M}$ pulls the cable.

We define two motion primitives for the dual arm robot to accomplish the task:

- *rotate*: $\mathcal{F}$ hold + $\mathcal{M}$ rotate.
- *pull*: $\mathcal{F}$ release + $\mathcal{M}$ pull.

## 6.7 Planner

In this section, we introduce planners for the robot motion. For each contact, the robot executes the *pull* motion primitive to reach an initial pose for making contact and then the *rotate* motion primitive to construct and use the contact. Since the cable length is much larger than the contact radius, the radius is assumed to be neglectable in the section.

The flow chart describing the overall motion generated by the planner for $n$ ordered contacts is shown in Fig. 6.8.

### 6.7.1 Initial Pose Planner

The initial pose planner plans a discrete target pose for the mobile end-effector $\mathcal{M}$ around each contact. We denoted the planned pose as $\boldsymbol{q}^* = [\boldsymbol{p}^{*T}\ \theta^*]^T$, with $\boldsymbol{p}^* = [x_{\boldsymbol{p}^*}\ y_{\boldsymbol{p}^*}]^T$ as position and $\theta^*$ as orientation.

Let us consider a general case in Fig. 6.9. We set the fixed point on the cable at $\boldsymbol{f} = [x_f\ y_f]^T$, the current contact at $\boldsymbol{c} = [x_c\ y_c]^T$, the next contact at $\boldsymbol{r} = [x_r\ y_r]^T$. The fixed point is either at the origin of $\mathcal{F}xy$ or the previous contact location which regulates the cable.

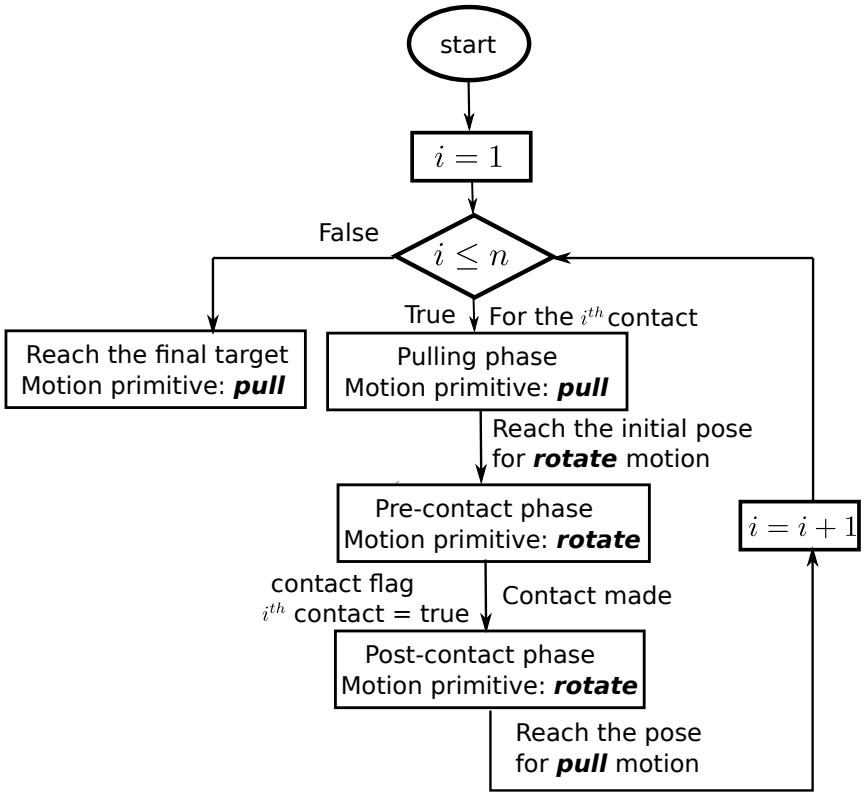



Figure 6.8: Flow chart depicting the steps needed to reach the final target by contact-based manipulation with $n$ contacts.

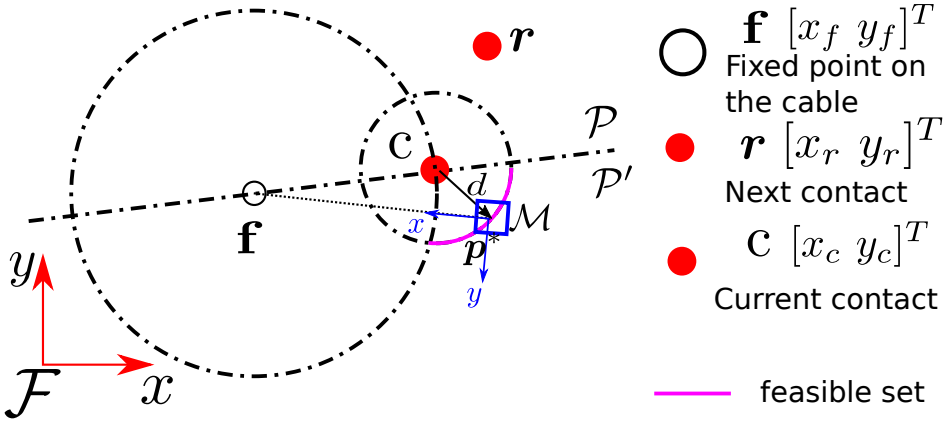



Figure 6.9: Position planning for a single contact.

After having reached the planned pose we *rotate* with the center at $\boldsymbol{f}$ to construct a contact, so the cable length is fixed. Thus, we have our first condition on $\boldsymbol{p}^*$:

$$\left\{\boldsymbol{p}^* \in \mathbb{R}^2 \;\middle|\; ||\overrightarrow{\boldsymbol{fp}^*}|| > ||\overrightarrow{\boldsymbol{fc}}||\right\}. \tag{6.7}$$

Otherwise, the cable cannot reach the contact by rotation.

We separate the plane into two half planes, by a line connecting the fixed point $\boldsymbol{f}$ and the contact $\boldsymbol{c}$. We denote the half plane with the next contact as $\mathcal{P}$ and the other one as $\mathcal{P}'$. To reach the next contact, we need:

$$\boldsymbol{p}^* \in \mathcal{P}'. \tag{6.8}$$

We envision $\mathcal{M}$ to be close the contact, because if it is far, more effort is needed to bring the cable in contact. To this end, we impose a third condition:

$$\left\{\boldsymbol{p}^* \in \mathbb{R}^2 \;\middle|\; ||\overrightarrow{\boldsymbol{cp}^*}|| = d\right\}, \tag{6.9}$$

where $||\overrightarrow{\boldsymbol{cr}}|| > d > 0$. In practical implementations, $d$ should be set considering the end-effector size (if too small, the end-effector may collide with the contact), and the camera field of view (large $d$ may jeopardize contact visibility).

By combining (6.7) - (6.9), we obtain a feasible set for $\boldsymbol{p}^*$:

$$\left\{\boldsymbol{p}^* \in \mathbb{R}^2 \;\middle|\; \mathcal{P}' \cap \left\{||\overrightarrow{\boldsymbol{fp}^*}|| > ||\overrightarrow{\boldsymbol{fc}}||\right\} \cap \left\{||\overrightarrow{\boldsymbol{cp}^*}|| = d\right\}\right\}, \tag{6.10}$$

which is illustrated in Fig. 6.9 (purple arc).

The local coordinate frame $\mathcal{M}xy$ should have its positive $x$ direction towards $\boldsymbol{f}$ (the center of rotation). Thus, the target orientation $\theta^*$ can be calculated as:

$$\begin{aligned} \boldsymbol{\Delta} &= \boldsymbol{f} - \boldsymbol{p}^*, \\ \theta^* &= \arctan \frac{y_{\boldsymbol{\Delta}}}{x_{\boldsymbol{\Delta}}} \end{aligned} \tag{6.11}$$

Extending the method to multi-contact cases is straightforward. For each contact, the fixed point $\boldsymbol{f}$ is at the previous contact (or at the origin of $\mathcal{F}xy$ for the first contact). For the last contact, the next contact $\boldsymbol{r}$ is at the target position. We calculate the planned pose for each contact in the given order. Figure 6.10 shows a planning example.

Once the planned pose is reached, we go to the pre-contact phase.

### 6.7.2 Pre-contact Phase

In the pre-contact phase, the local planner generates motions to reach the contact. It continuously receives contact detection information from a vision-based contact detector.

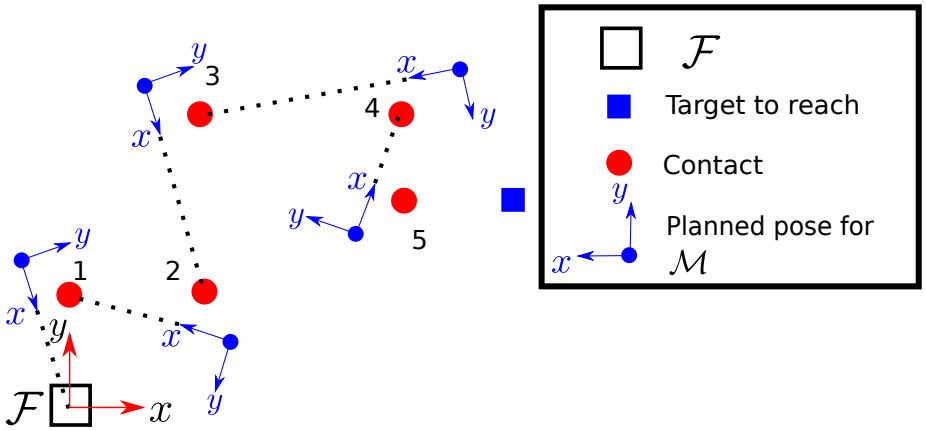



Figure 6.10: Multiple contacts planning example. The order of the contacts is presented by the numbering besides each contact, and known by the robot, which can then compute the initial pose.

When a contact occurs, the robot moves to the post-contact phase and the planner changes its behaviour.

In pre-contact phase, we *rotate* the cable to construct a contact. $\mathcal{F}$ will hold the cable so that the cable length is fixed. We denote a set of planned rotational positions as:

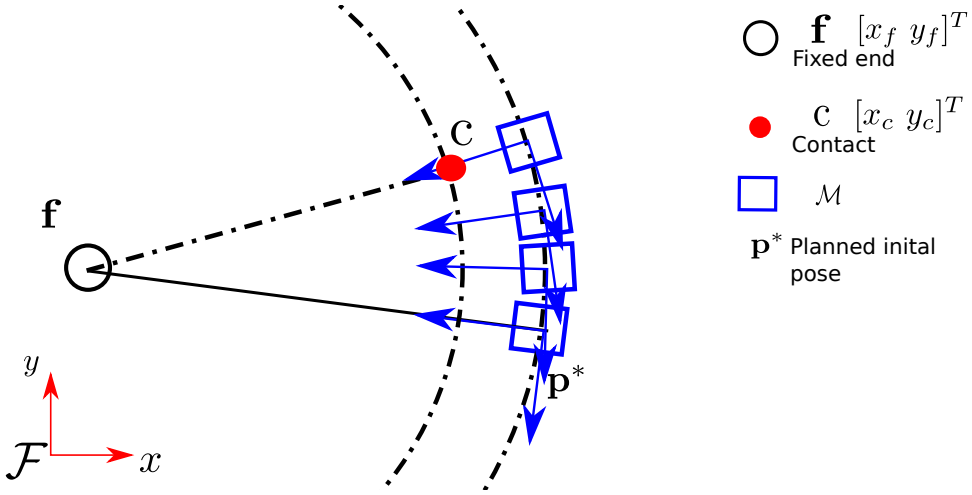



Figure 6.11: Rotation to reach the contact.

$$\boldsymbol{P} = \left[\boldsymbol{p}^1\ \boldsymbol{p}^2\ \cdots\right], \tag{6.12}$$

where $\boldsymbol{p}^i$ is the planned position on the manipulation plane:

$$\boldsymbol{p}^i = [x_p^i\ y_p^i]^T \in \mathbb{R}^2. \tag{6.13}$$

The rotational direction can be either clockwise or anti-clockwise. In Fig. 6.11, with

initial position of $\mathcal{M}$ at $\boldsymbol{p}^*$, we define the two vectors:

$$\boldsymbol{l}_1 = \boldsymbol{p}^* - \boldsymbol{f} \in \mathbb{R}^2,$$
$$\boldsymbol{l}_2 = \boldsymbol{c} - \boldsymbol{f} \in \mathbb{R}^2.$$

Using $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$, we can formulate a $2 \times 2$ matrix:

$$\boldsymbol{L} = [\boldsymbol{l}_1, \boldsymbol{l}_2] \in \mathbb{R}^{2\times 2}. \tag{6.14}$$

The rotational direction can be calculated by:

$$s = \text{sgn}(\det|\boldsymbol{L}|) \begin{cases} 1, & \text{rotate anti-clockwise,} \\ -1 & \text{rotate clockwise.} \end{cases} \tag{6.15}$$

The radius for the rotation is

$$r = ||\boldsymbol{p}^* - \boldsymbol{f}||. \tag{6.16}$$

We set the rotational step to be $\delta\theta$. For each $\boldsymbol{p}^i$ we set orientation $\theta_i = \theta^* + i\ \delta\ \theta s$. The position vector $\boldsymbol{p}^i = [x_p^i\ y_p^i]^T$ is then

$$\boldsymbol{p}^i = \boldsymbol{f} + r \begin{bmatrix} \cos\theta_i \\ -\sin\theta_i \end{bmatrix} \in \mathbb{R}^2. \tag{6.17}$$

The full pose vector for $\mathcal{M}$ is $[\boldsymbol{p}^{i^T}\ \theta_i]^T$. A planning example is shown on Fig. 6.11. $\mathcal{M}$ continues to *rotate* until contact occurs.

Since the contact is not on the robot but on the cable, contact forces cannot be directly measured on the robot. In addition, the contact force can be very small, which makes it hard to detect. Therefore, we use vision for the detection. We present the contact detector in Section 6.8.

### 6.7.3 Post-contact Phase

Once a contact is made, the robot uses the contact to shape the cable, until a good configuration is obtained for reaching the next planned initial pose (or to reach the target). When a contact occurs, the robot has to adapt its manipulation behaviour. The example overall motion is shown in Fig. 6.12. Let us denote the position of $\mathcal{M}$ at the time of contact by $\boldsymbol{p}'$. In the post-contact phase, contact $\boldsymbol{c} = [x_c\ y_c]^T$ becomes the new fixed point, which is also the new center for rotation. The new rotation radius is:

$$r' = ||\boldsymbol{p}' - \boldsymbol{c}||. \tag{6.18}$$

We re-plan the robot motion with the new center $\boldsymbol{c}$ and radius $r'$ in the same manner

as in the pre-contact phase, and keep the rotational direction. As discussed in Section 6.5, this post-contact rotation enlarges the contact curve, so the ACMI decreases. This way, the robot utilizes the contact for shaping the cable, and the contact is strengthened by the motion.

From the initial planner we get the position of the next planned pose $\boldsymbol{t}$. End-effector $\mathcal{M}$ continues to *rotate* until it lies on the line connecting $\boldsymbol{c}$ and $\boldsymbol{t}$. Then, the robot *pulls* the cable towards $\boldsymbol{t}$ (see Fig. 6.12). Since we consider the radius of the circular contact neglectable ($r \ll l$), then in Fig. 6.13 we have:

$$\overrightarrow{\boldsymbol{ce}} = \overrightarrow{\boldsymbol{c\rho}} + \overrightarrow{\boldsymbol{\rho e}} \approx \overrightarrow{\boldsymbol{\rho e}}. \tag{6.19}$$

The direction of the *pull* is the same as the sliding, which we analyzed in Section 6.5. Therefore the ACMI stays unchanged during the *pull*, the cable maintains contact.

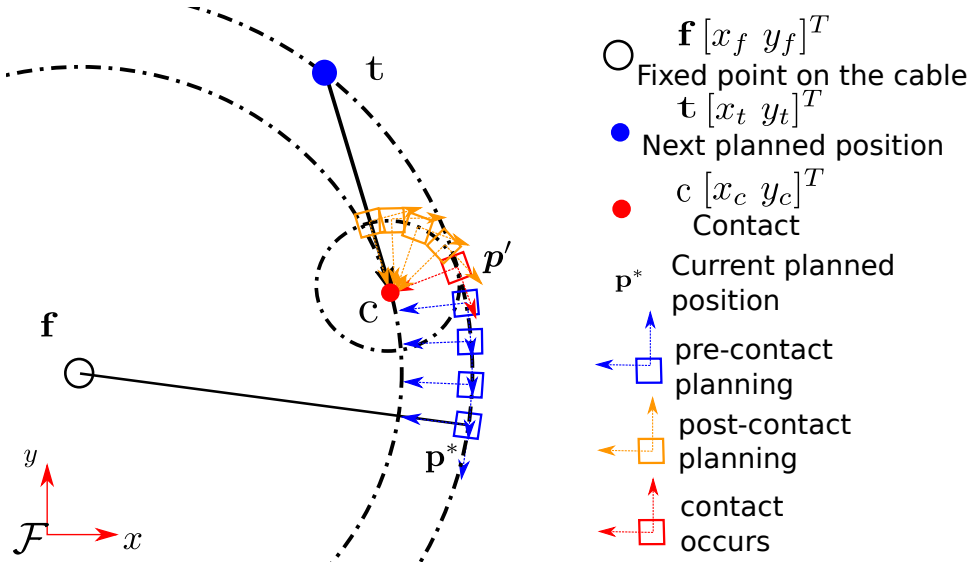



Figure 6.12: Full rotational motion planning.

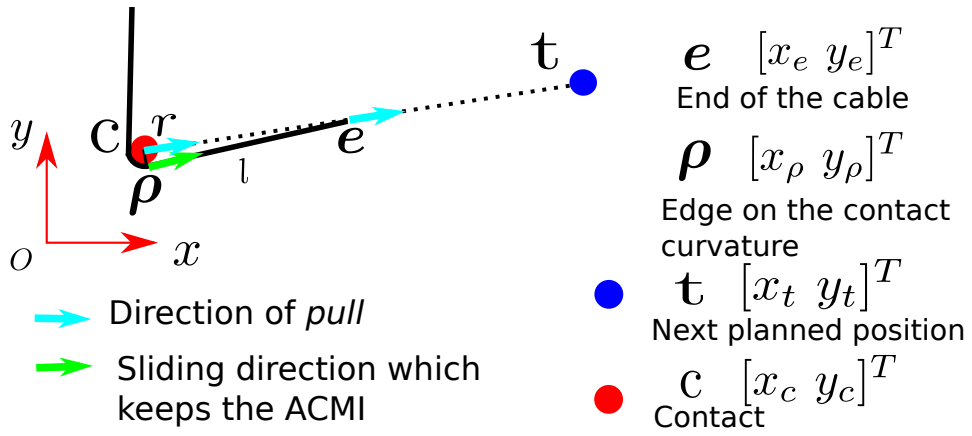



Figure 6.13: The *pull* can be regarded as a sliding motion.

## 6.8 Contact Detector

We detect the contact by extracting three features on the in-hand camera image, respectively: the locations of two ends of the cable segment and the contact. The detector is active only when contact is in the image. For the ease of extraction, we use a white manipulation plane, a black cable and blue contacts. OpenCV is used for image processing. Below is the procedure to extract these features:

- The blue contact can be found on the image with HSV thresholding. The contact location is the centroid of the biggest detected blob (Fig. 6.14b).
- The image is converted to grey-scale and then binarized using uniform thresholding. We use the Canny edge detector and then find contour to get the contours in the image. The contour is white in Fig. 6.14c.
- The relative position of the cable end held by $\mathcal{M}$ and the camera is fixed, therefore the hold end of the cable on the image can be found by finding the bottom point on the contour in a small Region of Interest (ROI). (Fig. 6.14c orange ROI).
- The other end of the cable can be found by setting another ROI at the top of the image. The upmost point on the contour within the ROI (green) can be regarded as the location of the other end. (Fig. 6.14c).

The feature extraction results are shown in Fig. 6.14d with the end hold by the end-effector $\mathcal{M}$ orange, the other end green, and the contact red.

Given the locations of the three features: orange $\boldsymbol{e}_o = [u_o\ v_o]^T$, green $\boldsymbol{e}_g = [u_g\ v_g]^T$, and red $\boldsymbol{e}_r = [u_r\ v_r]^T$ on the image, we define two vectors:

$$\boldsymbol{v}_{og} = \boldsymbol{e}_o - \boldsymbol{e}_g,$$
$$\boldsymbol{v}_{rg} = \boldsymbol{e}_r - \boldsymbol{e}_g.$$

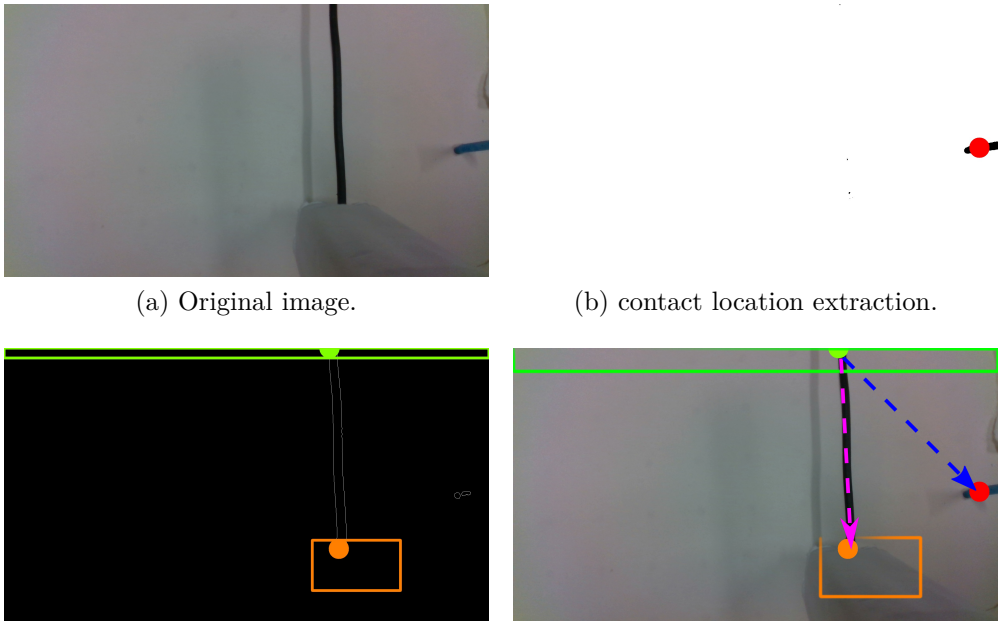
We detect the contact by setting a angular threshold $\delta\boldsymbol{v}$, and the contact is detected when:

$$|\angle(\boldsymbol{v}_{og}, \boldsymbol{v}_{rg})| < \delta\boldsymbol{v}.$$

## 6.9 Robotic Experiments

### 6.9.1 Hardware Setup

We use our BAZAR robot [Cherubini et al., 2019] which is equipped with two lightweight KUKA LWR IV. Planning is carried out in the task space, then projected in the joint space via inverse kinematics. To avoid kinematic singularities, we use adaptive damped



(a) Original image.

(b) contact location extraction.

(c) Contour of the cable, ROIs for cable end location extraction.

(d) Image after processing with locations of three features.

Figure 6.14: Extraction of the cable ends and the contact.

least squares [Chiaverini et al., 1991]. The trajectory generation is done by reflexxes motion generation library [Kröger, 2011].

A table serves as the manipulation plane. We use cylinder screws as contacts and a board with holes for easy insertion of contacts. A white wall paper covers the board, with only the contacts standing out. The size of the manipulation plane is 0.5 $m \times 0.9$ $m$.

Figure 6.15a shows the end-effector $\mathcal{M}$. A cable can be firmly attached at the bottom of it. An Intel Realsense D435 camera is mounted, with adjustable height from the bottom $(3 - 27cm)$. We only use the RGB images. The image resolution is $1920 \times 1080$. $\mathcal{F}$ is a 3D printed structure with 4 springs. When pressed on the table, it holds the cable; when lifted, it lets the cable slide (Fig. 6.15b).

Figure 6.16 shows the coordinate frames. An ArUco marker [Garrido-Jurado et al., 2016] is placed just below $\mathcal{F}$ to serve as the origin of the manipulation plane. The manipulation plane is parallel to $XY$ plane in the robot frame.

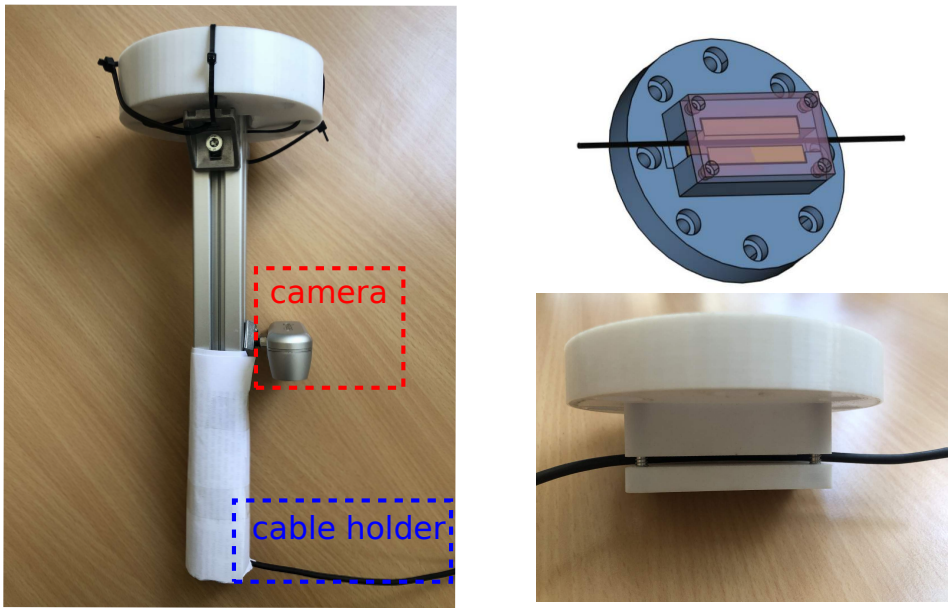
The transformation from any position $[X\ Y\ Z]^T \in \mathcal{R}XYZ$ to $[x\ y]^T \in \mathcal{F}xy$ is given by:

$$[x\ y]^T = [X\ Y]^T - [X_{\mathcal{F}}\ Y_{\mathcal{F}}]^T, \tag{6.20}$$

where $[X_{\mathcal{F}}\ Y_{\mathcal{F}}]^T$ is the $XY$ position of $\mathcal{F}$ in the robot frame.

### 6.9.2 Contact Localization

The location of contacts in the robot frame is obtained by commanding an initial upward motion of $\mathcal{M}$ to capture an image of the manipulation plane with the ArUco marker.




(a) End-effector $\mathcal{M}$. (b) End-effector $\mathcal{F}$.

Figure 6.15: Designs of the two end-effectors.

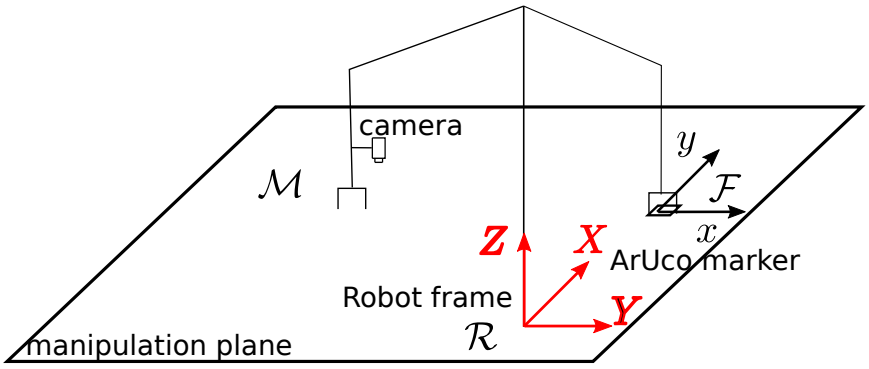



Figure 6.16: Setup and coordinate frames.

Using this image, with simple HSV thresholding, we find the blue contact locations on the image plane. The projection equation from 3D to 2D is then:

$$[u\ v\ 1]^T = \boldsymbol{DCT}_{\mathcal{R}}^{\mathcal{C}}[X\ Y\ Z\ 1]^T, \tag{6.21}$$

where $\boldsymbol{D} \in \mathbb{R}^{3\times3}$, $\boldsymbol{C} \in \mathbb{R}^{3\times4}$ are respectively distortion and camera matrices, and $\boldsymbol{T}_{\mathcal{R}}^{\mathcal{C}} \in \mathbb{R}^{4\times4}$ is the transformation from the robot to the camera frame. Both the depth $Z$ and $\boldsymbol{T}_{\mathcal{R}}^{\mathcal{C}}$ can be obtained via ArUco. Given image coordinates $u$ and $v$, the projection equation (6.21) consists of two functions with two unknowns. Thus, we solve the robot frame coordinate $X$ and $Y$ from $u$ and $v$. The contacts' order are given a prior to the robot.

### 6.9.3 Results

To validate our framework, we did 8 experiments with different contact configurations. The video of the experiment can be found at `http://y2u.be/7CdNQ4R_wT0`. Figure 6.17 shows the manipulation time for each experiment scenario[2]. The majority of time is dedicated to the rotation, during which the contact detector is active. We use a rotation step of 0.02 rad. The rotation step is chosen based on our image processing rate (on average 60 milliseconds per image) for an accurate contact detection. A late detection will put the cable in high tension. With faster image processing, we could reduce the execution time in the rotation phase by taking larger rotation steps. This could be achieved – for instance – by using smaller size images.

Figure 6.21 presents graphically all the multi-contact scenarios with nominal cable configurations. In the figure, the contact and initial/target $XY$ ($Z$ is constant) locations are given by vectors (in meters) expressed in the robot frame $\mathcal{R}XYZ$. The initial pose of $\mathcal{M}$ is $[-0.05\ 0.65\ \pi]^T$, and we set the target orientation for $\mathcal{M}$ as $\pi$.

Figure 6.18 shows the final configuration for these scenarios. The robot achieves all these configurations by contact regulation. Figure 6.22 shows a step by step manipulation process for scenario 8.

We can analyze the effect of robot motion on the ACMI for individual contact and then for the overall experiment. For an experiment with $n$ contacts, the initial overall ACMI is $2\pi n$. We expect the ACMI to decrease as the cable is regulated by each contact. For instance, the initial overall ACMI for scenario 8 is $3 \times 2\pi = 6\pi$. Given the motion in Fig. 6.19a, we can calculate the ACMI for the $i^{th}$ contact as shown on Fig. 6.19b:

$$\text{ACMI}_i = 2\pi - \pi - \alpha_i - \beta_i. \tag{6.22}$$

The $-\pi$ in (6.22) comes from the contact detected by the vision-based detector. We

[2]We developed a website indicating all the control and vision parameters used in our experiments: `https://jihong-zhu.github.io/robotics/2019/08/17/Experiment-contact-based-manipulation.html`.

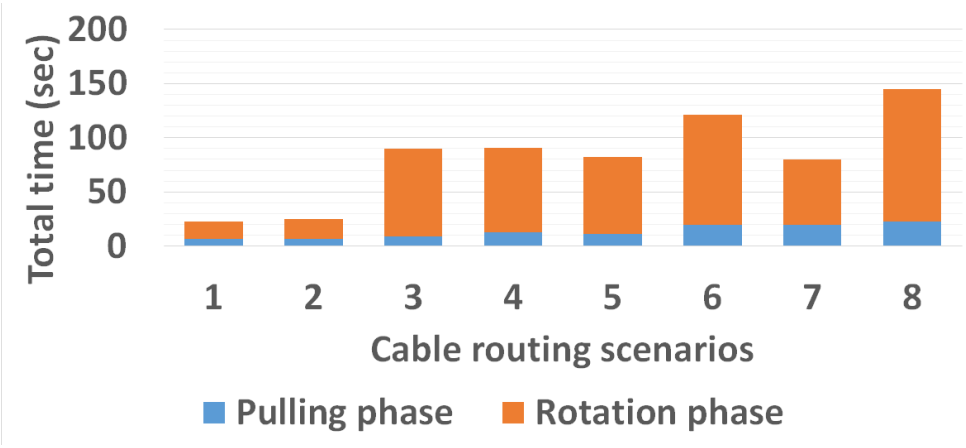
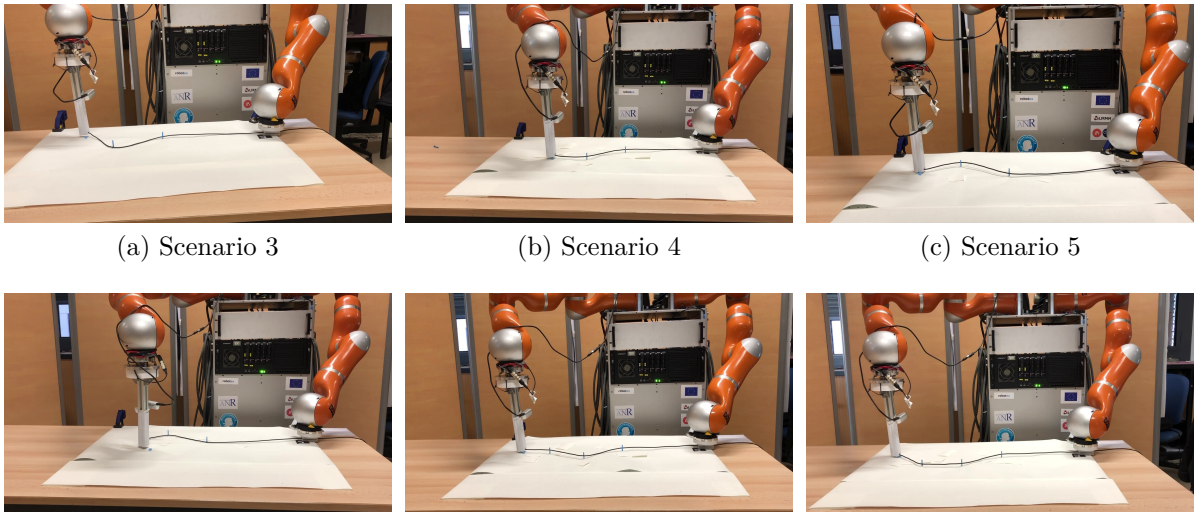


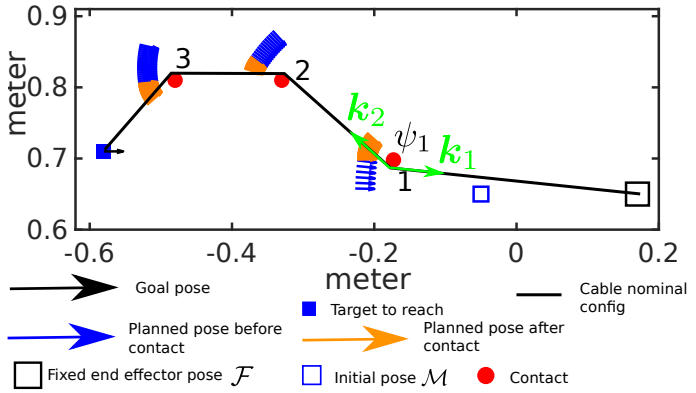

Figure 6.17: Total manipulation time for each scenario. Single contact cases: 1,2. Two contacts cases: 3-5. Three contacts cases: 6-8.



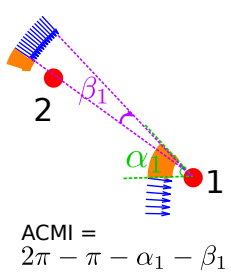
(a) Scenario 3 (b) Scenario 4 (c) Scenario 5

(d) Scenario 6 (e) Scenario 7 (f) Scenario 8

Figure 6.18: Final cable configurations in six of the eight scenarios.




(a) Planned motion for scenario 8




(b) ACMI calculation.

Figure 6.19: (a): Planned motion for scenario 8; (b): Example calculation of the ACMI of contact 1 after the robot motion.

consider a point contact is constructed. The $\alpha_i > 0$ is the rotation angle subsequent to the contact detection. The $\beta_i \in [-\pi, \pi]$ is the rotation angle after *pull* until the next contact is detected; $\beta_i$ is positive if the rotation after the *pull* has the same direction as before the *pull*, and negative otherwise. For the last contact $\beta_n = 0\,\text{rad}$.

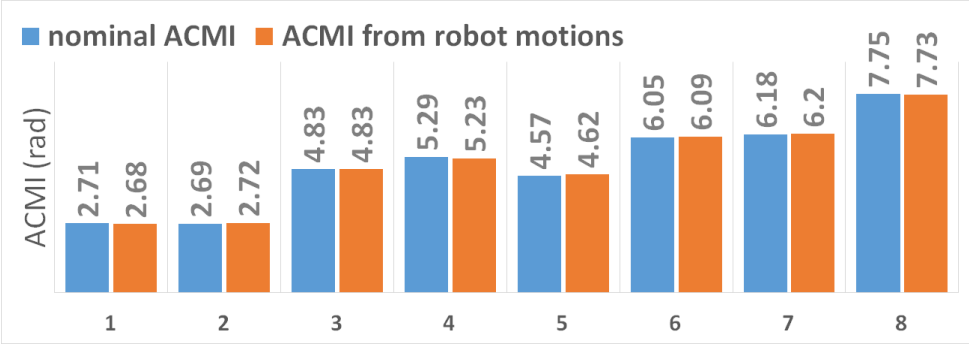



Figure 6.20: The nominal ACMI and the ACMI after the manipulation.

For the example on Fig. 6.19b, from the motion and contact locations we are able to calculate $\alpha_1 = 0.76\,\text{rad}$ and $\beta_1 = -0.0862\,\text{rad}$ ($\beta$ is negative as the direction is different from the rotation direction before the *pull*). The ACMI for contact is: $2\pi - \pi - 0.76 + 0.086 = 2.46\,\text{rad}$. ACMIs for the rest of contacts can be calculated in the same manner.

One simple measure to analyse overall contact mobility is by summing up the ACMI of each contact for a given scenario. The total ACMI for a $n$ contacts setup after robotic manipulation can be calculated by (6.23):

$$\mathcal{T} = \sum_{i=1}^{n} \text{ACMI}_i. \tag{6.23}$$

For a given topology of contact locations (order known), and the start and target end-effector positions, one could calculate the nominal total ACMI by the cable configuration. Consider the example in Fig. 6.19, based on the nominal cable configuration (obtained by connecting the start, contacts in order and the target sequentially), and the nominal ACMI for the first contact is $\psi_1$ which is the angle between the $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ (see Fig. 6.19). Similarly, using (6.23) we can calculate total nominal ACMI for a given the topology. Figure 6.20 presents a bar graph of comparison between the nominal and actual ACMI in each scenario. The difference between the nominal ACMI and ACMI calculated from the robot motion is very small. This confirms that with our framework the robot is able to construct contact and to use it for shaping the cable to reach the desired configuration.

## 6.10 Conclusion

The work is a pioneer research which considers environmental contacts in deformable object manipulation. In the robot experiments, we were constrained by the operational

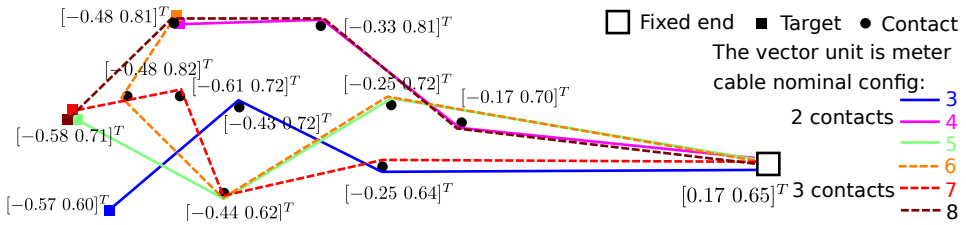
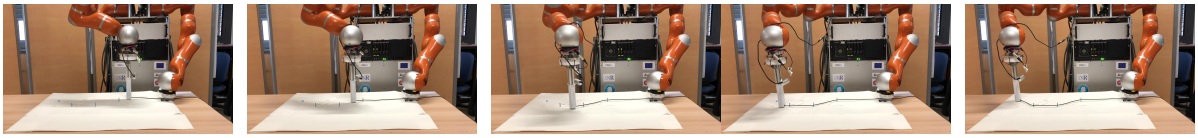



Figure 6.21: Manipulation experiments with more than one contact. Contacts are denoted with black dots, and the nominal cable configuration is drawn with solid (2 contacts) and dashed (3 contacts) lines



(a) Starting pose. (b) First contact. (c) Second contact. (d) Third contact. (e) Reach the final pose.

Figure 6.22: Manipulation process for scenario 8.

space of the robot, and by the simple mechanical structure designed to hold the cable. To extend the framework for more contacts one needs to consider: 1. re-grasping of the cable, 2. using a mobile base to enlarge the robot operational space. We believe contacts play a vital role in deformable objects manipulation. Currently, since we consider a robot with fixed base, the number of contacts is limited, as is the robot operational space. With a mobile base, one could enhance the robot operational range. In addition, since the rationale behind a human using a specific contact for shaping is probably closely related to cognitive science, some inspiration can be drawn from [Tee et al., 2018] for studying how robots should use different contacts. In a nutshell, contact for deformable objects manipulation is a rich area with a lot of new research opportunities. We hope this work can be a starting point for future research.

Chapter 7

# Conclusion and Future Work

In this thesis, we have worked on robotic manipulation of DLOs using visual feedback. First, we have extended the Fourier-based feedback method from [Navarro-Alarcon and Liu, 2017] to work with a dual arm setup and open contours. Taking a step further, we have proposed a novel scheme for manipulation of both deformable and rigid objects alike. Finally, inspired by humans using contacts for manipulation, we have introduced a framework that allows a dual arm robot to use environmental contacts for shaping DLOs.

## 7.1 Summary

We contribute to the state-of-the-art in robotic manipulation with both new algorithms and applications. On the algorithm side, our manipulation scheme unifies the manipulation of rigid and deformable objects. On the application side, our planning framework utilizes environmental contacts to make a dual arm robot capable of executing cable routing tasks.

We started by extending a model-free Fourier-based shape servoing scheme. In order to validate our approach before the robotic experiment, we worked on modeling DLOs to simulate our approach. The Fourier-based parameterization is general but requires an unnecessarily large number of features.

We continued our research in seeking a generalized method for vision-based robotic manipulation that works for both deformable and rigid objects. We decomposed the manipulation as two sub-tasks, namely parameterization and control. By keeping the model-free setup, the control is done with a local linear interaction matrix. To be consistent with the control, we introduced the corresponding linear parameterization method PCA. This consistency in parameterization and control allowed us to investigate the properties of this framework using fundamental theories in linear algebra. The resulting scheme was validated on simulation, and with robotic manipulation of rigid and deformable objects.

Another area of research in this thesis considers environmental contacts for manipulation. Instead of avoiding contact, we proposed a framework that utilizes them for manipulation. The target application was cable routing with circular contacts. We introduced an index namely ACMI to quantify the task executio. A vision-based contact detector aided the planner by switching the planning behavior once contact was detected.

With the proposed framework, the robot can perform cable routing tasks that have never been done in previous robotic research.

We have demonstrated an initial step to generalize rigid and deformable object manipulation in one unified framework. On the other line of research, we have conducted pioneering research on using environmental contacts for deformable object manipulation.

## 7.2 Limitation and Future Work

The contributions of this thesis is in the field of vision-based robotic manipulation. In the following, we briefly discuss the limitations of the presented work and future lines of research.

As we discussed in Chapter 5, the main limitations in both our model-free approaches on shape servoing is the proof of global convergence. In Section 5.6, We referred to it as the dilemma of a global deformation model. We do not explore 3D manipulation. If 3D perception and object state estimation are robust enough, our methods could be extended to 3D manipulation.

In contact-based deformable object manipulation, practically, the robot workspace is limited as the base is fixed. A better and more reliable vision pipeline is the key to integrate the control and planning under a vision-based framework. We should also investigate whether contacts can be detected by other sensing modalities such as force. On the theoretical side, our framework considers only DLOs with circular contacts, which limits the application to the execution of only one kind of contact-based tasks.

We give the future work based on the limitation mentioned above. Note that the open problems in the section are in the context of deformable object manipulation.

**3D Manipulation:** One of the most recent and promising works [Hu et al., 2018] implemented 3D manipulation with a dual arm. However, the motion is limited to 3D translations only. On multi-finger hand manipulation, [Ficuciello et al., 2018] proposed manipulation of 3D objects with a FEM-based deformation model. Nevertheless, this initial work was open-loop controlled. The complexity of 3D manipulation lies in both state estimation and control. In the state estimation, vision is the primary tool. However, it is subjected to light conditions and occlusions. On the control side, the problem is under-actuated in nature for the infinite DoFs in object deformation.

**Learning Simultaneously Local and Global Mapping:** To address the dilemma of a global deformation model, we could consider a two-layered hierarchical scheme. A fast and local manipulation scheme enables manipulation with minimum initialization. At the higher level, we could have a neural network based nonlinear function approximator for learning the global mapping. In this way, we are simultaneously computing a local and global model of the object. The combination is promising in resolving the issue of convergence in the local model while minimizing the undesired offline training for using

a global deformation model before starting the control.

**Multi-modal Manipulation:** The integration of other sensing modalities such as force, thermal (for considering deformation due to temperature) will be beneficial for better state estimation of the object. Force gives crucial information on deformation, which is currently lacking in most of the research. For plastic deformation, such as manipulating a clothe or a rope, the force can detect dangerous action that might tear the object apart. For elastic deformation, force thresholding may prevent damaging the elasticity of the object.

**Generalized Contacts:** Another interesting direction is the generalization of contacts in manipulation. One may find inspiration from cognitive science [Lewis, 2006,Säljö, 2002]. Works in affordance [Bærentsen and Trettvik, 2002, Jones, 2003] might also shed light on how to generalize the use of contacts for deformable object manipulation.

# Bibliography

[Acker and Henrich, 2003] Acker, J. and Henrich, D. (2003). Manipulating deformable linear objects: characteristic features for vision-based detection of contact state transitions. pages 204–209.

[Andersson et al., 2019] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36.

[Askeland and Phule, 2003] Askeland, D. R. and Phule, P. P. (2003). *The Science and Engineering of Materials*. Springer.

[Augugliaro et al., 2015] Augugliaro, F., Zarfati, E., Mirjan, A., and D'Andrea, R. (2015). Knot-tying with flying machines for aerial construction. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5917–5922. IEEE.

[Bærentsen and Trettvik, 2002] Bærentsen, K. B. and Trettvik, J. (2002). An activity theory approach to affordance. In *Proceedings of the second Nordic conference on Human-computer interaction*, pages 51–60. ACM.

[Bakthavatchalam et al., 2013] Bakthavatchalam, M., Chaumette, F., and Marchand, E. (2013). Photometric moments: New promising candidates for visual servoing. In *2013 IEEE Int. Conf. on Robotics and Automation*, pages 5241–5246. IEEE.

[Bakthavatchalam et al., 2018] Bakthavatchalam, M., Tahri, O., and Chaumette, F. (2018). A direct dense visual servoing approach using photometric moments. *IEEE Transactions on Robotics*, 34(5):1226–1239.

[Barr, 1987] Barr, A. H. (1987). Global and local deformations of solid primitives. In *Readings in Computer Vision*, pages 661–670. Elsevier.

[Bartels et al., 1987] Bartels, R., Beatty, J., and Barsky, B. (1987). *An Introduction to Splines for Use in Computer Graphics & Geometric Modeling*. Morgan Kaufmann Publishers Inc.

[Benítez and Montáns, 2017] Benítez, J. M. and Montáns, F. J. (2017). The mechanical behavior of skin: Structures and models for the finite element analysis. *Computers & Structures*, 190:75–107.

[Berenson, 2013] Berenson, D. (2013). Manipulation of deformable objects without modeling and simulating deformation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4525–4532. IEEE.

[Berkley et al., 1999] Berkley, J., Weghorst, S., Gladstone, H., Raugi, G., Berg, D., and Ganter, M. (1999). Fast finite element modeling for surgical simulation. *Studies in health technology and informatics*, 62:55–61.

[Bersch et al., 2011] Bersch, C., Pitzer, B., and Kammel, S. (2011). Bimanual robotic cloth manipulation for laundry folding. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1413–1419. IEEE.

[Borum and Bretl, 2015] Borum, A. and Bretl, T. (2015). The free configuration space of a kirchhoff elastic rod is path-connected. In *2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2958–2964. IEEE.

[Breen et al., 1994] Breen, D. E., House, D. H., and Wozny, M. J. (1994). Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 365–372. ACM.

[Bretl and McCarthy, 2014] Bretl, T. and McCarthy, Z. (2014). Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations. *The Int. Journal of Robotics Research*, 33(1):48–68.

[Brock and Khatib, 2002] Brock, O. and Khatib, O. (2002). Elastic strips: A framework for motion generation in human environments. *The Int. Journal of Robotics Research*, 21(12):1031–1052.

[Brown et al., 2004] Brown, J., Latombe, J.-C., and Montgomery, K. (2004). Real-time knot-tying simulation. *The Visual Computer*, 20(2-3):165–179.

[Broyden, 1965] Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation*, 19(92):577–593.

[Callister et al., 2007] Callister, W. D., Rethwisch, D. G., et al. (2007). *Materials science and engineering: an introduction*, volume 7. John wiley & sons New York.

[Celniker and Gossard, 1991] Celniker, G. and Gossard, D. (1991). Deformable curve and surface finite-elements for free-form shape design. *ACM SIGGRAPH Computer Graphics*, 25(4):257–266.

[Chadwick et al., 1989] Chadwick, J. E., Haumann, D. R., and Parent, R. E. (1989). Layered construction for deformable animated characters. In *ACM Siggraph Computer Graphics*, volume 23, pages 243–252. ACM.

[Chaumette, 2004] Chaumette, F. (2004). Image moments: a general and useful set of features for visual servoing. *IEEE Trans. on Robotics*, 20(4):713–723.

[Chaumette and Hutchinson, 2006a] Chaumette, F. and Hutchinson, S. (2006a). Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90.

[Chaumette and Hutchinson, 2006b] Chaumette, F. and Hutchinson, S. (2006b). Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90.

[Chaumette and Hutchinson, 2007a] Chaumette, F. and Hutchinson, S. (2007a). Visual servo control, part ii: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118.

[Chaumette and Hutchinson, 2007b] Chaumette, F. and Hutchinson, S. (2007b). Visual servo control, part II: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118.

[Chen, 1991] Chen, D. T.-W. (1991). *Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method*. PhD thesis, Massachusetts Institute of Technology.

[Cherubini et al., 2018] Cherubini, A., Leitner, J., Ortenzi, V., and Corke, P. (2018). Towards vision-based manipulation of plastic materials. In *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE.

[Cherubini et al., 2020] Cherubini, A., Ortenzi, V., Cosgun, A., Lee, R., and Corke, P. (2020). Model-free vision-based shaping of deformable plastic materials. *The Int. Journal of Robotics Research (to appear)*.

[Cherubini et al., 2019] Cherubini, A., Passama, R., Navarro, B., Sorour, M., Khelloufi, A., Mazhar, O., Tarbouriech, S., Zhu, J., Tempier, O., Crosnier, A., et al. (2019). A collaborative robot for the factory of the future: Bazar. *The International Journal of Advanced Manufacturing Technology*, pages 1–17.

[Chi and Berenson, ] Chi, C. and Berenson, D. Occlusion-robust deformable object tracking without physics simulation. In *2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[Chi and Berenson, 2019] Chi, C. and Berenson, D. (2019). Occlusion-robust deformable object tracking without physics simulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Chiaverini et al., 1991] Chiaverini, S., Egeland, O., and Kanestrom, R. K. (1991). Achieving user-defined accuracy with damped least-squares inverse kinematics. In *5th International Conference Advanced Robotics*, pages 672–677 vol.1.

[Collewet and Marchand, 2011] Collewet, C. and Marchand, E. (2011). Photometric visual servoing. *IEEE Trans. on Robotics*, 27(4):828–834.

[Collewet et al., 2008] Collewet, C., Marchand, E., and Chaumette, F. (2008). Visual servoing set free from image processing. In *2008 IEEE Int. Conf. on Robotics and Automation*, pages 81–86. IEEE.

[Coquillart, 1990] Coquillart, S. (1990). *Extended free-form deformation: a sculpturing tool for 3D geometric modeling.*

[Coquillart and Jancene, 1991] Coquillart, S. and Jancene, P. (1991). *Animated free-form deformation: an interactive animation technique*, volume 25.

[De Luca et al., 2006] De Luca, A., Albu-Schaffer, A., Haddadin, S., and Hirzinger, G. (2006). Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1623–1630.

[Deguchi and Noguchi, 1996] Deguchi, K. and Noguchi, T. (1996). Visual servoing using eigenspace method and dynamic calculation of interaction matrices. In *Proc. of 13th IEEE Int. Conf. on Pattern Recognition.*

[Delgado et al., 2015] Delgado, A., Jara, C. A., Mira, D., and Torres, F. (2015). A tactile-based grasping strategy for deformable objects' manipulation and deformability estimation. In *2015 12th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 369–374. IEEE.

[Drimus et al., 2014] Drimus, A., Kootstra, G., Bilberg, A., and Kragic, D. (2014). Design of a flexible tactile sensor for classification of rigid and deformable objects. *Robotics and Autonomous Systems*, 62(1):3–15.

[Etzmuß et al., 2003] Etzmuß, O., Keckeisen, M., and Straßer, W. (2003). A fast finite element solution for cloth modelling. In *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.*, pages 244–251. IEEE.

[Feller, 2008] Feller, W. (2008). *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons.

[Ficuciello et al., 2018] Ficuciello, F., Migliozzi, A., Coevoet, E., Petit, A., and Duriez, C. (2018). Fem-based deformation control for dexterous manipulation of 3d soft objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4007–4013. IEEE.

[Foresti and Pellegrino, 2004] Foresti, G. L. and Pellegrino, F. A. (2004). Automatic visual recognition of deformable objects for grasping and manipulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(3):325–333.

[Garrido-Jurado et al., 2016] Garrido-Jurado, S., noz Salinas, R. M., Madrid-Cuevas, F., and Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481 – 491.

[Guo et al., 2013] Guo, F., Lin, H., and Jia, Y.-B. (2013). Squeeze grasping of deformable planar objects with segment contacts and stick/slip transitions. In *IEEE Int. Conf. on Robotics and Automation*, pages 3736–3741. IEEE.

[Henrich et al., 1999] Henrich, D., Ogasawara, T., and Worn, H. (1999). Manipulating deformable linear objects-contact states and point contacts. In *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning, 1999.(ISATP'99)*, pages 198–204. IEEE.

[Hirai et al., 2001] Hirai, S., Tsuboi, T., and Wada, T. (2001). Robust grasping manipulation of deformable objects. In *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century.(Cat. No. 01TH8560)*, pages 411–416. IEEE.

[Hosoda and Asada, 1994] Hosoda, K. and Asada, M. (1994). Versatile visual servoing without knowledge of true jacobian. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, volume 1, pages 186–193. IEEE.

[Howard and Bekey, 2000] Howard, A. M. and Bekey, G. A. (2000). Intelligent learning for deformable object manipulation. *Autonomous Robots*, 9(1):51–58.

[Hu et al., 2019] Hu, Z., Han, T., Sun, P., Pan, J., and Manocha, D. (2019). 3-d deformable object manipulation using deep neural networks. *IEEE Robotics and Automation Letters*, 4(4):4255–4261.

[Hu et al., 2018] Hu, Z., Sun, P., and Pan, J. (2018). Three-dimensional deformable object manipulation using fast online gaussian process regression. *IEEE Robotics and Automation Letters*, 3(2):979–986.

[Huang et al., 2008] Huang, J., Di, P., Fukuda, T., and Matsuno, T. (2008). Dynamic modeling and simulation of manipulating deformable linear objects. In *2008 IEEE International Conference on Mechatronics and Automation*, pages 858–863. IEEE.

[Huang et al., 2005] Huang, J., Todo, I., and Yabuta, T. (2005). Position/force hybrid control of a manipulator with a flexible tool using visual and force information. In *Cutting Edge Robotics*. IntechOpen.

[Inoue, 1984] Inoue, H. (1984). Hand-eye coordination in rope handling. In *Proc. of Int. Symposium on Robotics Research*, pages 163–174. MIT PRESS.

[Jagersand et al., 1997] Jagersand, M., Fuentes, O., and Nelson, R. (1997). Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Proceedings of International Conference on Robotics and Automation*, volume 4, pages 2874–2880. IEEE.

[Jones, 2003] Jones, K. S. (2003). What is an affordance? *Ecological psychology*, 15(2):107–114.

[Kaboli et al., 2016] Kaboli, M., Yao, K., and Cheng, G. (2016). Tactile-based manipulation of deformable objects with dynamic center of mass. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 752–757. IEEE.

[Kallem et al., 2007] Kallem, V., Dewan, M., Swensen, J. P., Hager, G. D., and Cowan, N. J. (2007). Kernel-based visual servoing. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1975–1980. IEEE.

[Kang and Wen, 2002] Kang, H. and Wen, J. T. (2002). Robotic knot tying in minimally invasive surgeries. In *IEEE/RSJ International conference on intelligent robots and systems*, volume 2, pages 1421–1426. IEEE.

[Kheddar et al., 2019] Kheddar, A., Caron, S., Gergondet, P., Comport, A., Tanguy, A., Ott, C., Henze, B., Mesesan, G., Englsberger, J., Roa, M. A., Wieber, P., Chaumette, F., Spindler, F., Oriolo, G., Lanari, L., Escande, A., Chappellet, K., Kanehiro, F., and Rabate, P. (2019). Humanoid robots in aircraft manufacturing: The airbus use cases. *IEEE Robotics Automation Magazine*, 26(4):30–45.

[King et al., 2009] King, C.-H., Culjat, M. O., Franco, M. L., Lewis, C. E., Dutson, E. P., Grundfest, W. S., and Bisley, J. W. (2009). Tactile feedback induces reduced grasping force in robot-assisted surgery. *IEEE Transactions on Haptics*, 2(2):103–110.

[Kröger, 2011] Kröger, T. (2011). Opening the door to new sensor-based robot applications the reflexxes motion libraries. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*.

[Ladd and Kavraki, 2004] Ladd, A. M. and Kavraki, L. E. (2004). Using motion planning for knot untangling. *The International Journal of Robotics Research*, 23(7-8):797–808.

[Lagneau et al., 2020] Lagneau, R., Krupa, A., and Marchal, M. (2020). Active deformation through visual servoing of soft objects. In *IEEE Int. Conf. on Robotics and Automation*.

[Lamiraux and Kavraki, 2001] Lamiraux, F. and Kavraki, L. E. (2001). Planning paths for elastic objects under manipulation constraints. *The Int. Journal of Robotics Research*, 20(3):188–208.

[Lapresté et al., 2004] Lapresté, J.-T., Jurie, F., Dhome, M., and Chaumette, F. (2004). An efficient method to compute the inverse jacobian matrix in visual servoing. In *IEEE Int. Conf. on Robotics and Automation*.

[Laranjeira et al., 2017] Laranjeira, M., Dune, C., and Hugel, V. (2017). Catenary-based visual servoing for tethered robots. In *2017 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 732–738. IEEE.

[Laranjeira et al., 2020] Laranjeira, M., Dune, C., and Hugel, V. (2020). Catenary-based visual servoing for tether shape control between underwater vehicles. *Ocean Engineering*, 200:1–19.

[Lee et al., 2015] Lee, A. X., Lu, H., Gupta, A., Levine, S., and Abbeel, P. (2015). Learning force-based manipulation of deformable objects from multiple demonstrations. In *IEEE Int. Conf. on Robotics and Automation*. IEEE.

[Lewis, 2006] Lewis, J. W. (2006). Cortical networks related to human use of tools. *The neuroscientist*, 12(3):211–231.

[Li et al., 2011] Li, P., Lee, S.-h., and Hsu, H.-Y. (2011). Review on fruit harvesting method for potential use of automatic fruit harvesting systems. *Procedia Engineering*, 23:351–366.

[Luo and Nelson, 2001] Luo, Y. and Nelson, B. J. (2001). Fusing force and vision feedback for manipulating deformable objects. *Journal of Robotic Systems*, 18(3):103–117.

[Maitin-Shepard et al., 2010] Maitin-Shepard, J., Cusumano-Towner, M., Lei, J., and Abbeel, P. (2010). Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *IEEE Int. Conf. on Robotics and Automation*.

[Marchand, 2019] Marchand, E. (2019). Subspace-based direct visual servoing. *IEEE Robotics and Automation Letters*, 4(3):2699–2706.

[Miller, 1988] Miller, G. S. (1988). The motion dynamics of snakes and worms. In *ACM Siggraph Computer Graphics*, volume 22, pages 169–173. ACM.

[Moll and Kavraki, 2006] Moll, M. and Kavraki, L. E. (2006). Path planning for deformable linear objects. *IEEE Trans. on Robotics*, 22(4):625–636.

[Moore and Molloy, 2007] Moore, P. and Molloy, D. (2007). A survey of computer-based deformable models. In *Int. Machine Vision and Image Processing Conf.*, pages 55–66. IEEE.

[Murasugi, 2007] Murasugi, K. (2007). *Knot theory and its applications*. Springer Science & Business Media.

[Nair et al., 2017] Nair, A., Chen, D., Agrawal, P., Isola, P., Abbeel, P., Malik, J., and Levine, S. (2017). Combining self-supervised learning and imitation for vision-based rope manipulation. In *IEEE Int. Conf. on Robotics and Automation*, pages 2146–2153. IEEE.

[Navarro-Alarcon et al., 2019] Navarro-Alarcon, D., Cherubini, A., and Li, X. (2019). On model adaptation for sensorimotor control of robots. In *Chinese Control Conference*.

[Navarro-Alarcon and Liu, 2013] Navarro-Alarcon, D. and Liu, Y. (2013). Uncalibrated vision-based deformation control of compliant objects with online estimation of the jacobian matrix. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4977–4982.

[Navarro-Alarcon et al., 2013a] Navarro-Alarcon, D., Liu, Y., Romero, J. G., and Li, P. (2013a). Model-free visually servoed deformation control of elastic objects by robot manipulators. *IEEE Trans. on Robotics*, 29(6):1457–1468.

[Navarro-Alarcon et al., 2013b] Navarro-Alarcon, D., Liu, Y., Romero, J. G., and Li, P. (2013b). Visually servoed deformation control by robot manipulators. In *IEEE Int. Conf. on Robotics and Automation*, pages 5259–5264. IEEE.

[Navarro-Alarcon et al., 2014] Navarro-Alarcon, D., Liu, Y., Romero, J. G., and Li, P. (2014). On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments. *Int. Journal of Robotics Research*, 33(11):1462–1480.

[Navarro-Alarcon and Liu, 2014] Navarro-Alarcon, D. and Liu, Y.-h. (2014). A dynamic and uncalibrated method to visually servo-control elastic deformations by fully-constrained robotic grippers. In *IEEE Int. Conf. on Robotics and Automation*, pages 4457–4462. IEEE.

[Navarro-Alarcon and Liu, 2017] Navarro-Alarcon, D. and Liu, Y.-H. (2017). Fourier-based shape servoing: A new feedback method to actively deform soft objects into desired 2-D image contours. *IEEE Trans. on Robotics*.

[Navarro-Alarcon et al., 2016] Navarro-Alarcon, D., Yip, H. M., Wang, Z., Liu, Y., Zhong, F., Zhang, T., and Li, P. (2016). Automatic 3-D manipulation of soft objects by robotic arms with an adaptive deformation model. *IEEE Trans. on Robotics*, 32(2):429–441.

[Nayar et al., 1996] Nayar, S. K., Nene, S. A., and Murase, H. (1996). Subspace methods for robot vision. *IEEE Trans. on Robotics and Automation*, 12(5):750–758.

[Nealen et al., 2006] Nealen, A., Müller, M., Keiser, R., Boxerman, E., and Carlson, M. (2006). Physically based deformable models in computer graphics. In *Computer graphics forum*, volume 25, pages 809–836. Wiley Online Library.

[Newman and Newman, 2015] Newman, B. M. and Newman, P. R. (2015). *Theories of human development*. Psychology Press.

[Phillips et al., 2002] Phillips, J., Ladd, A., and Kavraki, L. E. (2002). Simulated knot tying. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 1, pages 841–846. IEEE.

[Platt and Badler, 1981] Platt, S. M. and Badler, N. I. (1981). Animating facial expressions. In *ACM SIGGRAPH computer graphics*, volume 15, pages 245–252. ACM.

[Qin et al., 2019] Qin, Y., Escande, A., and Yoshida, E. (2019). Cable installation by a humanoid integrating dual-arm manipulation and walking. In *IEEE/SICE Int. Symposium on System Integration (SII)*, pages 98–103. IEEE.

[Rimon and Burdick, 1998] Rimon, E. and Burdick, J. W. (1998). Mobility of bodies in contact. II. how forces are generated by curvature effects. *IEEE Transactions on Robotics and Automation*, 14(5):709–717.

[Roa and Suárez, 2015] Roa, M. A. and Suárez, R. (2015). Grasp quality measures: review and performance. *Autonomous robots*, 38(1):65–88.

[Roussel and Taïx, 2014] Roussel, O. and Taïx, M. (2014). Deformable linear object manipulation planning with contacts. In *Robot Manipulation: What has been achieved and what remains to be done? Full day workshop at IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

[Säljö, 2002] Säljö, R. (2002). Learning as the use of tools: a sociocultural perspective on the human–technology link. In *Learning with computers*, pages 158–175. Routledge.

[Sang and Tao, 2012] Sang, Q. and Tao, G. (2012). Adaptive control of piecewise linear systems: the state tracking case. *IEEE Trans. Autom. Control*, 57(2):522–528.

[Schein and Elber, 2004] Schein, S. and Elber, G. (2004). Discontinuous free form deformations. In *12th Pacific Conference on Computer Graphics and Applications*, pages 227–236. IEEE.

[Sederberg and Parry, 1986] Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. pages 151–160.

[Shirai and Inoue, 1973] Shirai, Y. and Inoue, H. (1973). Guiding a robot by visual feedback in assembling tasks. *Pattern recognition*, 5(2):99–108.

[Smith et al., 1996] Smith, P. W., Nandhakumar, N., and Ramadorai, A. K. (1996). Vision based manipulation of non-rigid objects. In *IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 3191–3196. IEEE.

[Strang, 1993] Strang, G. (1993). *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA.

[Takamatsu et al., 2006] Takamatsu, J., Morita, T., Ogawara, K., Kimura, H., and Ikeuchi, K. (2006). Representation for knot-tying tasks. *IEEE Transactions on Robotics*, 22(1):65–78.

[Tee et al., 2018] Tee, K. P., Li, J., Chen, L. T. P., Wan, K. W., and Ganesh, G. (2018). Towards emergence of tool use in robots: Automatic tool recognition and use without prior tool learning. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 6439–6446. IEEE.

[Tu and Terzopoulos, 1994] Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 43–50. ACM.

[Vaillant et al., 2016] Vaillant, J., Kheddar, A., Audren, H., Keith, F., Brossette, S., Escande, A., Bouyarmane, K., Kaneko, K., Morisawa, M., Gergondet, P., et al. (2016). Multi-contact vertical ladder climbing with an hrp-2 humanoid. *Autonomous Robots*, 40(3):561–580.

[Wakamatsu and Hirai, 2004] Wakamatsu, H. and Hirai, S. (2004). Static modeling of linear object deformation based on differential geometry. *The Int. Journal of Robotics Research*, 23(3):293–311.

[Wakamatsu et al., 1995] Wakamatsu, H., Hirai, S., and Iwata, K. (1995). Modeling of linear objects considering bend, twist, and extensional deformations. In *IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 433–438.

[Wang et al., 2018] Wang, Z., Li, X., Navarro-Alarcon, D., and Liu, Y.-h. (2018). A unified controller for region-reaching and deforming of soft objects. In *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 472–478. IEEE.

[Waters, 1987] Waters, K. (1987). A muscle model for animation three-dimensional facial expression. *ACM SIGGRAPH Computer Graphics*, 21(4):17–24.

[Yamaguchi and Atkeson, 2016] Yamaguchi, A. and Atkeson, C. G. (2016). Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables. In *IEEE-RAS 16th Int. Conf. on Humanoid Robots (Humanoids)*, pages 1045–1051. IEEE.

[Yoshida et al., 2015] Yoshida, E., Ayusawa, K., Ramirez-Alpizar, I. G., Harada, K., Duriez, C., and Kheddar, A. (2015). Simulation-based optimal motion planning for deformable object. In *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*.

[Yue and Henrich, 2002] Yue, S. and Henrich, D. (2002). Manipulating deformable linear objects: sensor-based fast manipulation during vibration. In *IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 2467–2472. IEEE.

[Zhang et al., 2017] Zhang, T., Caron, S., and Nakamura, Y. (2017). Supervoxel plane segmentation and multi-contact motion generation for humanoid stair climbing. *Int. Journal of Humanoid Robotics*, 14(01).

[Zhu et al., 2018] Zhu, J., Navarro, B., Fraisse, P., Crosnier, A., and Cherubini, A. (2018). Dual-arm robotic manipulation of flexible cables. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[Zhu et al., 2020a] Zhu, J., Navarro, B., Passama, R., Fraisse, P., Crosnier, A., and Cherubini, A. (2020a). Robotic manipulation planning for shaping deformable linear objects withenvironmental contacts. *IEEE Robotics and Automation Letters*, 5(1):16–23.

[Zhu et al., 2020b] Zhu, J., Navarro-Alarcon, D., Passama, R., and Cherubini, A. (2020b). Vision-based manipulation of deformable and rigid objects using subspace projections of 2d contours.

Appendix A

# Appendix

The appendix contains prove of **Lemma** 1 in Section 5.3.1, and also the equivalence of the solution (5.20) to the optimization problem we mentioned in Section 5.3.4.

## A.1 Prove of the Lemma

$\text{rank}(\boldsymbol{AB}) \leq \min(\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B}))$.

*Proof.* For ease of notation, we consider $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times k}$, and $\boldsymbol{C} = \boldsymbol{AB} \in \mathbb{R}^{m\times k}$. We write $\boldsymbol{B}$ into column vectors:

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_k \end{bmatrix} \tag{A.1}$$

We have:

$$\begin{aligned} \boldsymbol{C} &= \begin{bmatrix} \boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_k \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{Ab}_1, \boldsymbol{Ab}_2, \cdots, \boldsymbol{Ab}_k \end{bmatrix} \end{aligned} \tag{A.2}$$

The columns of $\boldsymbol{C}$ is linear combination of columns of $\boldsymbol{A}$, thus

$$\text{rank}(\boldsymbol{C}) = \text{rank}(\boldsymbol{AB}) \leq \text{rank}(\boldsymbol{A}). \tag{A.3}$$

Then, we write $\boldsymbol{A}$ into row vectors:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_m \end{bmatrix} \tag{A.4}$$

We have:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1\boldsymbol{B} \\ \boldsymbol{a}_2\boldsymbol{B} \\ \vdots \\ \boldsymbol{a}_m\boldsymbol{B} \end{bmatrix} \tag{A.5}$$

The rows of $\boldsymbol{C}$ is linear combination of rows of $\boldsymbol{B}$, thus

$$\text{rank}(\boldsymbol{C}) = \text{rank}(\boldsymbol{AB}) \leq \text{rank}(\boldsymbol{B}). \tag{A.6}$$

Combining (A.3) and (A.6), we have $\text{rank}(\boldsymbol{AB}) \leq \min(\text{rank}(\boldsymbol{A}), \text{rank}(\boldsymbol{B}))$. ■

## A.2 Solution to the Optimization Problem

Given $\boldsymbol{\Delta S}_i \in \mathbb{R}^{n\times M}$ and $\boldsymbol{\Delta R}_i \in \mathbb{R}^{n\times M}$, we would like to find $\boldsymbol{L}_i \in \mathbb{R}^{n\times n}$ given the equation:

$$\boldsymbol{\Delta S}_i = \boldsymbol{L}_i \boldsymbol{\Delta R}_i. \tag{A.7}$$

We can solve (A.7) by the optimization problem $\min_{\boldsymbol{L}_i}(\mathcal{J})$ with

$$\mathcal{J} = \sum_{j=1}^{n} ||\delta\boldsymbol{\sigma}_j^T - \Delta\boldsymbol{R}_i^T \boldsymbol{l}_j^T||_2, \tag{A.8}$$

where $\boldsymbol{\sigma}_j$ and $\boldsymbol{l}_j$ are respectively the $j^{th}$ row of $\Delta\boldsymbol{S}_i$ and $\boldsymbol{L}_i \in \mathbb{R}^{n\times n}$. The $||\cdot||_2$ denote the Euclidean norm. In this section, we want to show that the solution of the optimization problem is equivalent to (5.20), that is:

$$\hat{\boldsymbol{L}}_i = \boldsymbol{\Delta S}_i \boldsymbol{\Delta R}_i^T (\boldsymbol{\Delta R}_i \boldsymbol{\Delta R}_i^T)^{-1}. \tag{A.9}$$

We assume the same as in Section 5.3.4 that $\text{rank}(\boldsymbol{\Delta R}_i) = n$.

We can decompose the solving of $\min_{\boldsymbol{L}_i}(\mathcal{J})$ into solving individual sub-problems. We rewrite (A.8) into:

$$\min_{\boldsymbol{L}_i}(\mathcal{J}) = \min_{\boldsymbol{l}_1}(\mathcal{J}_1) + \min_{\boldsymbol{l}_2}(\mathcal{J}_2) + \ldots + \min_{\boldsymbol{l}_n}(\mathcal{J}_n), \tag{A.10}$$

where

$$\mathcal{J}_i = ||\delta\boldsymbol{\sigma}_j^T - \Delta\boldsymbol{R}_i^T \boldsymbol{l}_j^T||_2, \tag{A.11}$$

The individual optimization problem

$$\min_{\boldsymbol{l}_i}(\mathcal{J}_i) = \min_{\boldsymbol{l}_i} ||\delta\boldsymbol{\sigma}_j^T - \Delta\boldsymbol{R}_i^T \boldsymbol{l}_j^T||_2 \tag{A.12}$$

is a standard linear least square problem ($\boldsymbol{Ax} = \boldsymbol{b}$):

$$\Delta\boldsymbol{R}_i^T \boldsymbol{l}_j^T = \delta\boldsymbol{\sigma}_j^T. \tag{A.13}$$

Its solution is:

$$\boldsymbol{l}_j^T = (\boldsymbol{\Delta R}_i \boldsymbol{\Delta R}_i^T)^{-1} \boldsymbol{\Delta R}_i \delta\boldsymbol{\sigma}_j^T \tag{A.14}$$

Transposing (A.14) on both side, we have:

$$\boldsymbol{l}_j = \delta\boldsymbol{\sigma}_j\boldsymbol{\Delta R}_i^T(\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T)^{-1}, \tag{A.15}$$

Writing (A.15) in the matrix form:

$$\hat{\boldsymbol{L}}_i = \boldsymbol{\Delta S}_i\boldsymbol{\Delta R}_i^T(\boldsymbol{\Delta R}_i\boldsymbol{\Delta R}_i^T)^{-1}, \tag{A.16}$$

which is exactly (A.9).