Vision-based shape servoing of soft objects using the mass-spring model

Fouad Makiyeh

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : « *Automatique, Productique et Robotique* »

Par

« Fouad MAKIYEH »

« Vision-based shape servoing of soft objects using mass-spring model »

Thèse présentée et soutenue à Rennes, le 12/12/2023
Unité de recherche : « Centre Inria de l'Université de Rennes »

Rapporteurs avant soutenance :

Andrea CHERUBINI	Professeur des Universités, Université de Montpellier
David NAVARRO-ALARCON	Associate Professor, The Hong Kong Polytechnic University

Composition du Jury :

Président :	Christian DURIEZ	Directeur de recherche, Centre Inria de l'Université de Lille
Examinateurs :	Andrea CHERUBINI	Professeur des Universités, Université de Montpellier
	David NAVARRO-ALARCON	Associate Professor, The Hong Kong Polytechnic University
	Ekrem MISIMI	Senior Scientist, SINTEF Ocean (Norvège)
Dir. de thèse :	Alexandre KRUPA	Directeur de recherche, Centre Inria de l'Université de Rennes
Co-dir. de thèse :	François CHAUMETTE	Directeur de recherche, Centre Inria de l'Université de Rennes
Co-encad. de thèse :	Maud MARCHAL	Professeur des Universités, INSA de Rennes

Acknowledgement

I would like to express my heartfelt gratitude to the members of the jury for their insightful questions and valuable contributions during the defense of this thesis. Your expertise and engagement have added significant depth to the examination process. Special thanks are extended to the rapporteurs for their meticulous comments and constructive feedback on the manuscript, undoubtedly enhancing the quality of this work.

Genuine thanks go to Ekrem MISIMI for funding the GentleMan project and extending a warm invitation to Norway. Collaborating with Ekrem and the project members has been enriching, significantly contributing to the success of this research.

Deep gratitude is extended to Alexandre KRUPA, François CHAUMETTE, and Maud MARCHAL for their guidance and mentorship, instrumental in shaping the direction of this thesis. A special acknowledgment to Fabien SPINDLER for his consistent availability and support with any issues related to the robotic platform.

I would also like to express appreciation to colleagues and Inria members for their collaboration and camaraderie, which made this research endeavor both enriching and rewarding. Additional thanks are conveyed to Bangalore Ravi KIRAN for being a consistent pillar of support, inspiration, and motivation, as well as to my close friends who consistently make my days more enjoyable.

Special and profound thanks go to each member of my family, playing a unique and invaluable role in my academic journey. To my parents, your unwavering love, guidance, and sacrifices have been the bedrock of my perseverance and success, fueling my determination. To my siblings, your support and understanding have been a constant source of strength, making this journey more meaningful.

In conclusion, my gratitude extends to all who have contributed to the realization of this thesis. Your support has been invaluable.

SYNTHÈSE EN FRANÇAIS

Le domaine de la robotique a connu des avancées dans la manipulation de divers objets, permettant aux robots d'accomplir des tâches complexes. Par exemple, les robots agricoles sont capables de récolter des fruits et légumes, mais cette récolte pose un défi en raison du risque de destruction de ces produits si des forces de préhension excessives sont appliquées. Cette tâche représente un des nombreux cas où des robots entrent en contact avec des objets souples susceptibles de changer de forme. La manipulation d'objets déformables concerne des objets tels que des tissus, des câbles ou bien encore des organes humains qui peuvent subir des déformations non rigides. Cela nécessite des développements particuliers et des stratégies nouvelles pour une manipulation robotique efficace. En effet, la commande des robots pour saisir ou manipuler des objets souples est complexe, et les lois de commande classiques utilisées pour saisir des objets rigides ne peuvent plus être utilisées. Néanmoins, la capacité des robots à manipuler de tels objets souples peut aboutir à de nouvelles possibilités d'automatisation, à améliorer la productivité et renforcer la collaboration entre homme et robot dans divers domaines, notamment dans l'industrie pour la fabrication et l'emballage de produits, dans le domaine médical pour l'assistance aux gestes chirurgicaux et dans de nombreux autres secteurs.

Considérons par exemple le contexte de l'emballage du saumon. Dans ce scénario, le robot doit d'abord détecter le saumon, puis le saisir tout en maintenant sa forme d'origine ou en le déformant pour qu'il prenne la forme souhaitée, facilitant ainsi la tâche de découpe ultérieure. Les systèmes robotiques équipés d'outils de découpe avancés utilisent des algorithmes de découpe précis pour garantir des tailles de filets précises et uniformes, réduisant ainsi les coûts de main-d'œuvre, améliorant la productivité et préservant l'intégrité du produit. Le développement d'un système robotique pour le conditionnement du saumon constitue l'une des principales préoccupations du projet GentleMAN. Ma thèse de doctorat fait partie de ce projet, financé par le Conseil de la recherche de Norvège, dirigé par l'Institut Sintef à Trondheim, et impliquant comme partenaires le Laboratoire d'informatique et d'intelligence artificielle du MIT (États-Unis), l'Université de technologie du Queensland (Australie) et le centre Inria de l'Université de Rennes. Cette thèse se concentre donc sur la manipulation d'objets déformables à l'aide de robots, plus spé-

cifiquement de bras manipulateurs. De plus, cette recherche vise à exploiter les avancées en robotique et en vision par ordinateur pour développer des méthodologies innovantes et des stratégies de commande permettant aux robots de percevoir et de commander de manière adaptative la forme des objets déformables.

Défis

La manipulation d'objets déformables est un domaine en évolution avec divers défis que les chercheurs abordent activement, car ces objets posent d'importantes difficultés en termes de commande et de manipulation.

Tout d'abord, les interactions entre les bras robotiques et les objets déformables impliquent des incertitudes au niveau du contact et des forces de friction. Ces incertitudes peuvent entraîner une instabilité dans la saisie ou provoquer le glissement de l'objet.

Un autre défi réside dans la prédiction du comportement des objets souples. Les matériaux déformables présentent des caractéristiques non linéaires et variables dans le temps, ce qui rend difficile le développement de modèles précis. Une solution possible est d'utiliser des approches basées sur des algorithmes d'apprentissage automatique pour apprendre le comportement de ces objets. Une autre solution envisageable consiste à utiliser des modèles physiques utilisés pour représenter la dynamique de tels objets. Cependant, ces modèles peuvent être peu précis à moins que leurs paramètres soient parfaitement connus.

De plus, estimer la forme, la pose et les déformations internes des objets déformables est difficile en raison de leur nature complexe et non rigide. Les techniques de vision par ordinateur, telles que la reconstruction 3D et les algorithmes d'estimation de forme, peuvent être utilisées pour obtenir des représentations précises des objets déformables. De plus, l'incorporation de simulations basées sur des modèles physiques qui capturent les déformations et les interactions de ces objets peut améliorer les capacités de perception pour les tâches de manipulation.

En outre, la manipulation d'objets déformables nécessite des stratégies de commande robustes et adaptatives. Les techniques traditionnelles de manipulation d'objets rigides ne sont pas directement applicables en raison de la nature flexible des corps déformables. De nouvelles approches de commande qui tiennent compte des changements dynamiques de la forme et de la compliance de l'objet sont nécessaires. Ces méthodes impliquent l'incorporation de capteurs pour estimer et mettre à jour de manière continue les informations sur les propriétés de l'objet ou les conditions de l'environnement. Cela garantit que le

contrôleur peut s'adapter et répondre de manière appropriée au comportement non rigide de l'objet considéré.

En résumé, relever les défis de la manipulation d'objets déformables nécessite une approche multidisciplinaire. En utilisant des modèles basés sur les données, des modèles physiques, l'haptique, des stratégies de commande adaptatives et des techniques de perception avancées, il est possible d'améliorer la précision, la stabilité et l'efficacité de la manipulation d'objets déformables.

Objectifs et contributions

Cette thèse contribue à l'état de l'art de la manipulation d'objets déformables en introduisant de nouvelles lois de commande basées sur un modèle physique pour la manipulation d'objets souples en temps réel. Plus précisément, nous utilisons un modèle simple qui est le modèle masse-ressort. Bien que ce modèle puisse capturer les caractéristiques essentielles des objets déformables, il peut rencontrer des difficultés à reproduire avec précision le comportement réel de l'objet. Cependant, en incorporant des techniques avancées de perception visuelle en temps réel, il est possible de corriger les décalages entre le modèle et la dynamique réelle de l'objet.

Le manuscrit de la thèse suit la méthodologie suivante :

Dans le deuxième chapitre, nous présentons une revue de l'état de l'art qui inclut différentes approches utilisées pour modéliser un objet souple, estimer les paramètres des modèles physiques utilisés pour décrire sa dynamique, différentes lois de commande utilisées pour sa manipulation, et divers algorithmes employés pour le suivre.

Dans le troisième chapitre, nous présentons une nouvelle méthode basée sur le système masse-ressort pour estimer le déplacement de chaque point du modèle en fonction des mouvements qui lui sont appliqués. Nous désignons par les points manipulés les points du modèle où les mouvements sont appliqués. Nous commençons par établir la relation analytique qui relie les déplacements des points de l'objet aux mouvements successifs des points manipulés. Cette relation forme la base de notre contribution, construite sur le modèle masse-ressort en tenant compte du retard de propagation qu'il introduit. Nous formulons les déplacements des points dans un maillage 1D en réponse à une succession de mouvements appliqués à un ou deux points manipulés, puis nous généralisons cette formulation pour un maillage 3D impliquant plusieurs points manipulés. En se basant sur ces formes analytiques, nous introduisons une loi de commmande qui permet de po-

sitionner indirectement plusieurs points de l'objet à leur position souhaitée en agissant sur des points manipulés éloignés. Dans ce même chapitre, nous présentons des résultats de simulations qui démontrent la robustesse du système en boucle fermée proposé. Nous prenons en compte des facteurs tels que la résolution du maillage et les incertitudes des paramètres du modèle pour évaluer l'efficacité de la loi de commande. Nous présentons également une étude comparative entre l'approche proposée et deux méthodes de l'état de l'art qui n'utilisent pas de modèle.

- Une vidéo qui présente divers résultats de simulation pour la tâche de positionnement indirecte et qui montre également la robustesse du contrôleur proposé vis-à-vis des incertitudes des paramètres du modèle ainsi que les résultats de comparaison, est disponible ici : [Vidéo-positionnement-indirect].

Le quatrième chapitre se concentre sur la validation expérimentale de la tâche de positionnement indirect avec des objets déformables. Cette tâche repose sur la méthodologie introduite dans le chapitre précédent, où nous utilisons à la fois des approches basées sur la vision (suivi) et des approches basées sur le modèle (loi de commande) pour accomplir le positionnement indirect de points caractéristiques d'objets souples, modélisés par un modèle masse-ressort. L'objectif est de démontrer l'efficacité de cette approche par des expériences réelles. Une première étape consiste à créer un modèle approximé de l'objet souple, puisque celui-ci n'est pas connu à l'avance. Le modèle est ensuite constitué par un maillage 3D décrivant la géométrie de l'objet et par un système masse-ressort pour estimer son comportement physique. Cependant, ce modèle étant une approximation, il peut y avoir un décalage entre le modèle et l'objet réel. Pour corriger cet écart, l'objet est suivi en utilisant une caméra RGB-D, puis réaligné en appliquant des contraintes externes sur l'objet qui dépendent des mesures visuelles. Une fois que le modèle de l'objet est créé et que ses paramètres sont approximés, des résultats expérimentaux sont présentés pour évaluer l'efficacité de l'approche proposée. Cette approche d'asservissement visuel combine à la fois le retour visuel et le comportement dynamique de l'objet qui est prédit par le modèle.

- Les résultats du positionnement indirect d'un point de l'objet ont été présentés dans notre première publication [Makiyeh et al., 2022], avec une vidéo associée disponible ici : [Vidéo-ICARCV].

- Une autre vidéo présentant divers résultats expérimentaux pour le positionnement

indirect de deux points de l'objet peut également être trouvée ici : [Vidéo-positionnement-indirect].

Le cinquième chapitre présente deux méthodes de commande basées sur le système masse-ressort pour amener des objets déformables à une forme globale désirée. Ces méthodes visent à déformer le contour pour des objets 2D et la surface complète pour des objets 3D. Pour réduire la complexité de la forme, deux approches de réduction de dimension sont proposées : les descripteurs de Fourier et les moments 3D. Ces approches sont évaluées à l'aide de simulations et d'expériences pour démontrer leur efficacité dans la commande de la forme d'objets souples. L'objectif est de permettre l'application d'une forme désirée à ces objets en utilisant un nombre limité de points manipulés.

- Les résultats de notre approche de commande de la forme d'objets en 2D et 3D basée sur les descripteurs de Fourier ont été présentés dans notre deuxième publication [Makiyeh et al., 2023], avec une vidéo illustrant deux résultats expérimentaux disponibles ici : [Vidéo-IROS].

Le dernier chapitre offre une conclusion sur les approches proposées, ainsi qu'une discussion sur les perspectives à court et à long terme de la recherche future.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

In recent years, the field of robotics has witnessed remarkable advancements in the manipulation of various objects, enabling robots to perform a wide range of complex tasks. For example, in packaging and assembly lines, robots within manufacturing industries are frequently utilized to grasp objects such as boxes and bottles from one location, and to accurately place them in another. However, during such tasks, these entities can be deformed, altering their shapes and potentially causing them to slip from the robotic hand. Additionally, in agriculture, agricultural robots are capable of tasks such as picking vegetables, though picking tomatoes presents a challenge due to the risk of destruction if excessive gripping forces are applied. Furthermore, in the field of medicine, surgical robots are employed to manipulate surgical tools during operations. However, these tools come into contact with delicate human organs such as the liver, necessitating careful consideration of the applied force to prevent damage.

Automating the manipulation of deformable objects with a robot presents a significant challenge due to their non-rigid and highly variable nature. Deformable object manipulation involves items like fabrics, cables, and biological tissues that can undergo non-rigid deformations, requiring distinct skills and strategies for effective robotic handling. This challenge is especially prominent within the field of robotics, as interacting with deformable objects poses unique difficulties. Nonetheless, the capacity of robots to handle such soft objects can unlock novel possibilities for automation, improve productivity, and enhance human-robot collaboration across various domains, including manufacturing, healthcare, and search-and-rescue operations.

As an illustrative example, let us consider the context of salmon packaging. In this scenario, the robot needs to first detect the salmon, then grasp it while maintaining its original shape or reshaping it into the desired forms that facilitate the subsequent cutting task. Robotic systems equipped with advanced cutting tools utilize precise cutting

algorithms to ensure accurate and uniform fillet sizes, thereby reducing labor costs, enhancing productivity, and maintaining product integrity. An example of this pipeline is presented in Figure. 1.1, which is one of the main concerns of the GentleMAN (299757) project. My PhD is part of this project, funded by The Research Council of Norway, led by the Sintef Institute in Trondheim, Norway, and involving partners such as the MIT Computer Science and Artificial Intelligence Laboratory (USA), Queensland University of Technology (Australia), and the Inria center at University of Rennes.

This PhD research aims to address the fundamental challenges associated with deformable object manipulation using robotic manipulators. By leveraging cutting-edge advancements in robotics and computer vision, this study seeks to develop novel methodologies and control strategies that enable robots to perceive and to control the shape of deformable objects in an adaptive manner.

Figure 1.1: An example of automated robotic salmon fillet cutting, from [Einarsdóttir et al., 2022].

1.2 Challenges

Deformable object manipulation is a dynamic and evolving field with various challenges that researchers actively tackle, as these objects pose significant difficulties in terms of accurate control and manipulation.

Firstly, the interactions between robotic grippers and deformable objects involve uncertainty in terms of contact and friction. Deformable materials often exhibit complex contact behavior, such as sticking, sliding, and rolling. This uncertainty may result in grasping instability or object slippage.

Another challenge involves predicting the deformation behavior of soft objects. Indeed, deformable materials exhibit non-linear and time-varying characteristics, making it challenging to develop accurate models. One possible solution is to employ data-driven approaches, such as machine learning algorithms, to learn the behavior of deformable objects from real-world sensor data. By training models on a diverse set of examples, these algorithms can capture the underlying dynamics and improve manipulation performance. Another possible solution is to consider physics-based models. However, these models may not achieve high accuracy unless the exact physics parameters of the actual soft objet are known. To address this limitation, incorporating real-time sensing data into the models, the object deformations can be refined.

Addionally, estimating the shape, pose, and internal deformations of deformable objects is challenging due to their complex and non-rigid nature. Computer vision techniques, such as 3D reconstruction and shape estimation algorithms, can be employed to obtain accurate representations of deformable objects. In addition, incorporating physics-based simulation that capture the deformations and interactions of these objects can enhance perception capabilities for manipulation tasks.

Furthermore, deformable object manipulation requires robust and adaptive control strategies. Traditional rigid object manipulation techniques are not directly applicable due to the flexible nature of deformable bodies. Control approaches that account for the dynamic changes in object shape and compliance are needed. These methods involve incorporating sensor feedback to consistently estimate and update information about the object properties or the environment conditions. This ensures that the controller can adapt and respond appropriately to these changes.

In summary, addressing the challenges of deformable object manipulation requires a comprehensive approach. By leveraging data-driven models, physics-based models, tactile sensing, adaptive control strategies, and advanced perception techniques, it is possible to improve the accuracy, stability, and efficiency of manipulating deformable objects.

1.3 Thesis objectives and contributions

The objective of this thesis is to develop a physics-based controller for deformable object manipulation, focusing on real-time applications.

The research aims to investigate the effectiveness of using a simple model that is, the mass-spring model, for shape servoing of soft objects. While this simple model can capture the essential characteristics of deformable objects, it may struggle to accurately replicate the real object behavior. However, by incorporating advanced real-time perception techniques, it is possible to correct any discrepancies or drift between the model and the actual object dynamics. By doing so, this research aims to develop a computationally efficient, interpretable, and uncertainty-aware controller.

This thesis contributes to the state-of-the-art in soft object manipulation by introducing novel control laws for controlling the shape of deformable objects. Shape servoing involves driving specific points of the object to desired positions by deforming the object using robotic manipulators.

To achieve this, the following chapters are included:

- Chapter 2 presents a comprehensive literature review of relevant state-of-the-art. It includes different approaches for modeling a soft object, estimating the parameters of physics-based models, various control laws used for manipulating a soft object, and diverse algorithms employed for tracking it.

- Chapter 3 outlines a novel physics-based method for estimating the displacement of any model point in function of the applied motions on the manipulated points based on the mass-spring model. This method is then employed to design a control law for the deformation task, enabling indirect positioning of object points by acting on different points. Indeed, the proposed control law addresses the challenge of considering the inertia and flexible nature of deformable objects. This chapter also includes simulation results illustrating the success and robustness of the indirect positioning task using the proposed approach. Furthermore, the chapter compares the effectiveness of our approach in achieving the indirect positioning task with two state-of-the-art model-free methods.

  - A video presenting various simulation results for the indirect positioning task, including the robustness of the proposed controller against model parameters and the comparison results, can be found here: [Indirect-positioning-video]

- Chapter 4 is dedicated to validating the indirect positioning task introduced in Chapter 3 within an experimental setup, considering various soft objects. To accomplish this, we begin by outlining the different steps for reconstructing a model corresponding to an unknown soft object, where both its geometry and material properties are unknown. However, given that physics-based models are approximations of real object behavior, there may exist a gap between the object model deformation and its actual behavior. To overcome this challenge, we employ a method involving real-time tracking of the object during deformation. Whenever a drift is detected, we rectify the object model by applying external forces, thereby aligning it with the real deformation. Subsequently, we accomplish the indirect positioning task of one or multiple object points to their desired positions. This is achieved by employing our proposed method, which integrates now both vision-based (tracking process) and model-based (control law) approaches. The deformation is achieved using two robotic manipulators and a RGB-D camera.
  - The results of the indirect potioning of one feature point were presented in our first publication [Makiyeh et al., 2022], with an accompanying video available here: [ICARCV-video]
  - Another video showing various experimental results for the indirect positioning of two feature points can also be found here: [Indirect-positioning-video]
- Chapter 5 introduces two physics-based methods for controlling the complete shape of the soft object. The object shape is defined by the complete set of its surface points in the case of 3D objects and by its 2D contour when dealing with 2D objects. To represent the object shape effectively using a low-dimensional feature vector, we present two distinct approaches. The first approach employs Fourier descriptors, while the second utilizes 3D moments. For each of these approaches, we assess the corresponding proposed control law for the deformation task by conducting a comprehensive evaluation that combines both simulation and experimental results.
  - The results of our servoing approach based on Fourier descriptors were presented in our second publication [Makiyeh et al., 2023], with an accompanying video illustrating two experimental results available here: [IROS-video]
- Chapter 6 offers a conclusion on the proposed approaches, as well as discussing short term and long-term perspectives on future research.

Chapter 2

State Of The Art

Deformable object manipulation has emerged as a challenging yet promising field with applications spanning robotics, medical procedures, and virtual reality. Perceiving and manipulating objects with flexible and complex structures, such as cloth, soft tissue, or cables, require precise control strategies that account for their nonlinear and dynamic behaviors.

This research specifically concentrates on the shape servoing of soft objects using robotic manipulators. Over time, researchers have explored various control methods and techniques to address this challenge and enhance manipulation capabilities.

To manipulate soft objects, some approaches first approximate their physical behavior using physics-based models and then develop control laws to guide the robots in accomplishing the task. In these model-based approaches, object models are simulated to derive the control laws, which are then applied to the robots in either open-loop execution or within a closed-loop system using feedback measurements. Open-loop systems typically rely on highly accurate models, while closed-loop systems can adapt to models of varying accuracy, benefiting from the feedback loop to execute the task effectively. When using physics-based models, it is crucial to note that these models depend on physical parameters specific to the deformable object. If these parameters are unknown, they must be estimated. Additionally, these models are still approximations of the real behavior of the object, meaning that the simulated deformation may deviate from the actual deformation of the object. This deviation can be detected by tracking the object, and as a result, continuous correction of any deviations becomes essential if they occur.

In contrast, other approaches rely solely on visual feedback information without the need for complex modeling of soft objects to generate the corresponding control laws. These approaches, known as model-free approaches, leverage vision feedback to identify object relevant features, and formulate a control strategy for closed-loop execution. The robot adjusts its position to align its observed features with the desired ones, corresponding to the intended task. This approach is commonly referred to as a visual servoing

problem.

This chapter provides an overview of the current literature in deformable object manipulation, highlighting the strengths and limitations of existing approaches. By exploring the recent advancements, this review sets the stage for the subsequent chapters, which aim to propose novel model-based control strategies and algorithms to enhance the manipulation of deformable objects in various practical applications.

We start by providing an introduction to the basics of visual servoing in Section 2.1, a fundamental aspect of robotics control method that utilizes visual data from cameras to direct a robot motion. Then, in Section 2.2, we delve into the various techniques that have been investigated for simulating soft object deformations, which is essential for model-based shape servoing approaches. Following that, we explore in Section 2.3 various tracking methods aimed at minimizing the discrepancy between simulated object deformations and their real deformations. Additionally, we review in Section 2.4 different approaches for physics-based model parameter estimation, which is necessary for model-based shape servoing approaches. Furthermore, we explore in Section 2.5 a wide range of techniques that have been studied for shape servoing of soft objects, encompassing model-based and model-free approaches. Lastly, in Section 2.6, we contextualize this work in relation to the existing literature.

2.1 Visual servoing

This section presents fundamental principles of visual servoing, which is a technique that allows robots to interact with both rigid and deformable objects. Visual servoing control involves using visual information provided by a visual sensor to control robot movements. This visual feedback information can be obtained from a camera attached to a robot, which moves along with the robot (eye-in-hand scenario), or from a stationary camera observing the robot within the workspace (eye-to-hand scenario).

2.1.1 Basics of visual servoing

The reader can refer to [Chaumette & Hutchinson, 2008] for a detailed presentation of the basics of visual servoing. Visual servoing is a technique that converts visual measurements into a vector of n visual features, denoted as $\boldsymbol{s} \in \mathbb{R}^n$, enabling the robot to interact with its environment. The difference between the current value of this vector and the desired

value, $\boldsymbol{s}^*$, is used to calculate the task error, $\boldsymbol{e}$, as follows:

$$\boldsymbol{e} = \boldsymbol{s} - \boldsymbol{s}^* \tag{2.1}$$

The task error is then used to generate control laws to be applied to the robots in order to minimize its norm.

When the variations in $\boldsymbol{s}$ are solely influenced by the camera motion, then the relationship between the changes in the visual feature vector, $\dot{\boldsymbol{s}}$, and the camera velocity $\boldsymbol{v}_c = (\mathbf{v}_c, \boldsymbol{\omega}_c) \in \mathbb{R}^6$, is expressed using an interaction matrix denoted as $\boldsymbol{J_s}$. Here, $\mathbf{v}_c$ represents the instantaneous linear velocity vector, and $\boldsymbol{\omega}_c$ represents the instantaneous angular velocity vector. In case $\boldsymbol{s}^*$ remains constant, which induces $\dot{\boldsymbol{e}} = \dot{\boldsymbol{s}}$, the expression of the relation between $\dot{\boldsymbol{e}}$ and $\boldsymbol{v}_c$ is therefore as follows:

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{s}} = \boldsymbol{J_s}\boldsymbol{v}_c = \boldsymbol{J_e}\boldsymbol{v}_c \tag{2.2}$$

Typically, the error is chosen to exhibit an exponential decoupled decrease, represented as $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$, where $\lambda > 0$. This specific selection plays a crucial role in designing the camera control velocity, denoted as $\boldsymbol{v}_c$, which follows the formulation:

$$\boldsymbol{v}_c = -\lambda \widehat{\boldsymbol{J_e}}^{+} \boldsymbol{e} \tag{2.3}$$

with $\widehat{\boldsymbol{J_e}}^{+}$ the Moore-Penrose pseudo-inverse of $\widehat{\boldsymbol{J_e}}$, an approximation of $\boldsymbol{J_e}$.

Visual servoing can be classified into two distinct categories: Image-Based Visual Servoing (IBVS) and Pose-Based Visual Servoing (PBVS). The former leverages the image to extract various features, such as image points, image moments, and image Fourier descriptors, to name a few. On the other hand, the latter employs image data to estimate three-dimensional measurements, such as the pose of an object, which subsequently aids in the servoing process. In this thesis, our focus lies on IBVS since it is related to our chosen methodology.

2.1.2 Classical Image-Based visual servoing

When constructing the visual feature vector $\mathbf{s}$ from image points $p_1, p_2, \ldots, p_n$, the following procedure is employed. Each image point p_i corresponds to a pair of horizontal and vertical coordinates (X_i, Y_i) obtained from projecting 3D points P_i, with corresponding 3D coordinates (x_i, y_i, z_i), expressed in the camera Cartesian frame. The visual feature

vector $\mathbf{s}$ is then formed as $\mathbf{s} = (p_1, p_2, \ldots, p_n)$.

For instance, when $\boldsymbol{s}$ is simply constructed from one image point, i.e $\boldsymbol{s} = (X_1, Y_1)$, the interaction matrix is obtained as follows:

$$\boldsymbol{J_s} = \begin{bmatrix} \frac{-1}{z_1} & 0 & \frac{X_1}{z_1} & X_1 Y_1 & -(1+X_1^2) & Y_1 \\ 0 & \frac{-1}{z_1} & \frac{Y_1}{z_1} & 1+Y_1^2 & -X_1 Y_1 & -X_1 \end{bmatrix} \tag{2.4}$$

It should be noted that the matrix in (2.4) depends on the depth information of the point. Therefore, when using a RGB camera, the depth information is not available, and an approximation of this matrix needs to be utilized. Multiple approaches for either estimating the point depth or directly numerically estimating the interaction matrix are recalled in [Chaumette & Hutchinson, 2008]. We now describe two methods for numerically estimating such a Jacobian matrix, the Broyden algorithm [Broyden, 2000] and Least-Squares Minimization [de Mathelin & Lozano, 1999].

2.1.2.1 Numerical estimation of the feature Jacobian matrix using the Broyden algorithm

The Broyden algorithm is an iterative method used to numerically estimate the Jacobian matrix of a function f [Broyden, 2000]. It is particularly useful when the function is expensive to compute or when the analytical expression for the Jacobian is not available. The Broyden algorithm updates an approximation of the Jacobian matrix based on the changes in the function outputs and inputs.

Let consider a function $f : \mathbb{R}^m \to \mathbb{R}^n$, where m is the dimension of the input vector $\boldsymbol{q} \in \mathbb{R}^m$ and n is the dimension of the output $\boldsymbol{s} \in \mathbb{R}^n$. The goal is to estimate the Jacobian matrix $\boldsymbol{J_f}$, a $n \times m$ matrix, with its approximation denoted $\widehat{\boldsymbol{J_f}}$. The different steps to update the Jacobian matrix in the Broyden algorithm are as follows:

1. Initialization: Start by providing $\widehat{\boldsymbol{J_f}}$, an initial approximation to $\boldsymbol{J_f}$. This can be a numerical approximation or an identity matrix, depending on the problem and available information.

2. Evaluation: Initialize an initial guess for the inputs, evaluate the function f at the initial guess, and compute the corresponding outputs.

3. Iterative Steps: Perform the following steps iteratively at each iteration k until convergence:

- Compute the difference between the function output and the previous output:

$$\boldsymbol{\delta s} = f(\boldsymbol{q}_k) - \boldsymbol{s}_{k-1}$$

- Compute the difference between the current input and the previous input:

$$\boldsymbol{\delta q} = \boldsymbol{q}_k - \boldsymbol{q}_{k-1}$$

- Update the Jacobian matrix with an update speed α:

$$\widehat{\boldsymbol{J_f}}_k = \widehat{\boldsymbol{J_f}}_{k-1} + \alpha \frac{\boldsymbol{\delta s} - \widehat{\boldsymbol{J_f}}_{k-1}\boldsymbol{\delta q}}{\boldsymbol{\delta q}^T \boldsymbol{\delta q}} \boldsymbol{\delta q}^T$$

- Update the inputs:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k - \widehat{\boldsymbol{J_f}}_k^{+} f(\boldsymbol{q}_k)$$

- Update the function outputs:

$$\boldsymbol{s}_{k+1} = f(\boldsymbol{q}_{k+1})$$

- Check for convergence. If the change in the inputs and outputs is below a specified tolerance, terminate the iterations.

4. Convergence: Once the iterations have converged, the final approximation of the Jacobian matrix, $\widehat{\boldsymbol{J_f}}$, is obtained.

The Broyden update is computationally efficient, and supports incremental updates, making it well-suited for situations where the matrix or its inverse requires adaptation in response to changing data or conditions. However, this approach does have some limitations. For instance, it can be sensitive to numerical stability issues, particularly when dealing with ill-conditioned matrices. Furthermore, it does not guarantee global convergence, or even convergence at all. Indeed, its convergence relies on the choice of the initial guess.

2.1.2.2 Numerical estimation of the feature Jacobian using Least-Squares Minimization

The Least-Squares Minimization algorithm is another method used for the numerical estimation of the Jacobian matrix of a function f [de Mathelin & Lozano, 1999]. This algorithm involves formulating the problem as a least-squares minimization, where the goal is to find the Jacobian matrix that minimizes the sum of squared residuals between the actual function outputs and the estimated outputs based on the Jacobian.

Let consider again a function $f : \mathbb{R}^m \to \mathbb{R}^n$. Here is the step-by-step explanation of the Least-Squares Minimization algorithm for estimating the Jacobian matrix $\boldsymbol{J_f}$, by $\widehat{\boldsymbol{J_f}}$:

1. Initialization: Start by providing $\widehat{\boldsymbol{J_f}}$ an initial approximation to $\boldsymbol{J_f}$.
2. Evaluation: Initialize an initial guess for the inputs, evaluate the function f at the initial guess, and compute the corresponding outputs.
3. Iterative Steps: Perform the following steps iteratively at each iteration k until convergence:
   - Compute the difference between the function output and the previous output:
   $$\boldsymbol{\delta s} = f(\boldsymbol{q}_k) - \boldsymbol{s}_{k-1}$$
   - Compute the difference between the current input and the previous input:
   $$\boldsymbol{\delta q} = \boldsymbol{q}_k - \boldsymbol{q}_{k-1}$$
   - Update the Jacobian matrix approximation by minimizing the sum of squared residuals:
   $$\widehat{\boldsymbol{J_f}}_k = arg\,min \|\boldsymbol{\delta s} - \widehat{\boldsymbol{J_f}}\boldsymbol{\delta q}\|^2$$
   - Update the inputs:
   $$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k - (\widehat{\boldsymbol{J_f}}_k^T \widehat{\boldsymbol{J_f}}_k)^+ \widehat{\boldsymbol{J_f}}_k \boldsymbol{\delta s}$$
   - Update the function outputs:
   $$\boldsymbol{s}_{k+1} = f(\boldsymbol{q}_{k+1})$$
   - Check for convergence.

4. Convergence: Once the change in inputs and outputs is below a specified tolerance, the final approximation of the Jacobian matrix $\boldsymbol{J_f}$, $\widehat{\boldsymbol{J_f}}$, is obtained.

The Least-Squares Minimization method is generally more numerically stable than the Broyden rule, especially when dealing with ill-conditioned matrices. However, it can be more computationally intensive. Moreover, this method can be sensitive to outliers in the data, which can significantly impact the estimated Jacobian.

2.1.3 Conclusion

This section offered a brief overview of the fundamentals of IBVS, which is used to control objects interacting with rigid scene, such as tracking a moving target [Papanikolopoulos et al., 1993]. Furthermore, we introduced two potential approaches for estimating the IBVS Jacobian matrix. These numerical estimations are useful when components of this matrix cannot be obtained directly from analytical form. The Broyden update rule has found application in the field of robotics. For example, [Hosoda & Asada, 1994] applied this approach to estimate a Jacobian matrix, allowing for the convergence of features to desired values without the need for prior knowledge of the kinematic structure of the robot system. [Jägersand & Nelson, 1996] utilized this technique within the context of visually-guided grasping. [Shahamiri & Jagersand, 2005] proposed an UIBVS method, an IBVS approach without the knowledge of camera or robot calibration, to achieve positioning tasks while avoiding obstacles. Regarding the Least-Squares Minimization algorithm, [Shademan et al., 2010] employed a robust local least-squares (LLS) technique for the control of a robot in a UIBVS context and for Jacobian estimation through Iteratively Reweighted Least Squares. The LLS method introduced by [Massoud Farahmand et al., 2007] enables the estimation of the Jacobian for any point from visual information within a nearby region of the point in question.

While these approaches were initially designed for robots interacting with rigid objects, which is not the primary focus of this thesis, some model-free methods have successfully adapted classical visual servoing principles for deformable object shape servoing. These methods will be further detailed in Section 2.5.

The next section presents principles of continuum mechanics and various approaches used to discretize object models, which are necessary for model-based approaches to manipulate deformable objects.

2.2 Deformation modeling

Continuum mechanics offers essential insights into the modeling and understanding of deformable objects. Deformable objects encompass a wide range of materials, from common substances like metals and polymers to biological tissues and geological formations, all of which can undergo deformation under various external forces.

Continuum mechanics provides a comprehensive framework for modeling the behavior of deformable materials in a continuous manner, treating materials as continuous media and encompassing concepts such as stress, strain, elasticity, and the partial differential equations governing their behavior. Therefore, this approach allows for highly accurate modeling of soft objects. To enable practical simulations of soft objects, numerical approximations based on continuum mechanics, such as the Finite Element Method (FEM) and mass-spring models (MSMs), have been developed. These methods facilitate efficient numerical solutions while introducing some level of approximation. The choice of method depends on the specific problem, the required level of accuracy, available computational resources, and whether real-time simulations are necessary.

In the subsequent sections, we will begin in Section 2.2.1 by providing a brief introduction to continuum mechanics, followed in Section 2.2.2, Section 2.2.3 and Section 2.2.4 by an exploration of various discretization methods used to approximate continuum mechanics. These methods play a crucial role in bridging the gap between the continuous mathematical descriptions of deformable objects and their practical computational implementation. Moreover, they showcase diverse levels of physical accuracy and simplicity, which are crucial considerations for the implementation of basic scenarios and adaptation to handle increasingly complex situations.

2.2.1 Overview on continuum mechanics

Continuum mechanics provides a framework for modeling deformable objects. Deformation refers to the alteration in shape or size of the object when subjected to external forces or loads. Deformation can be categorized as either elastic (reversible) or plastic (irreversible), depending on whether the object returns to its original shape after the forces are removed.

At time t_0, we consider the object in its rest state, not subject to any deformation. Let consider a point P_i belonging to the object material. Its corresponding coordinates

at each time (t), $\boldsymbol{x}_{i(t)}$, and velocities $\dot{\boldsymbol{x}}_{i(t)}$, are given by:

$$\boldsymbol{x}_{i(t)} = (x_{i(t)}, y_{i(t)}, z_{i(t)})$$
$$\dot{\boldsymbol{x}}_{i(t)} = (\dot{x}_{i(t)}, \dot{y}_{i(t)}, \dot{z}_{i(t)})$$

Then, when external forces are exerted on the object, it will deform in response to these forces, causing its points to move to new positions corresponding to the deformed configuration. Let $\boldsymbol{u}_i$ denote the displacement of P_i, given by:

$$\boldsymbol{u}_{i(t)} = (u_{ix}, u_{iy}, u_{iz}) = \boldsymbol{x}_{i(t)} - \boldsymbol{x}_{i(t_0)} \tag{2.5}$$

An example of this deformation is represented in Figure 2.1.

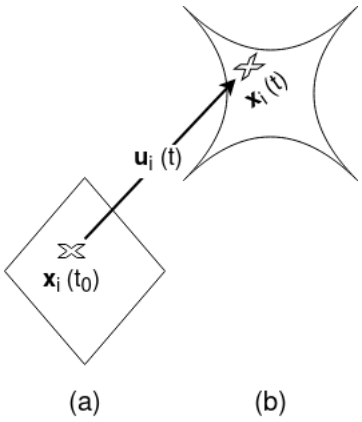


Figure 2.1: Illustration of a deformation with the displacement vector $\boldsymbol{u}_i(t)$: (a) and (b) correspond to the undeformed object and deformed object, respectively.

The object response to external displacements depends on its geometrical and physical properties. The deformation resulting from this displacement can be calculated by considering the stress tensor ($\boldsymbol{\sigma}$), which represents the force applied per unit area of the object shape, and the strain ($\boldsymbol{\epsilon}$), which quantifies the ratio of deformation to the original size of the object shape. Furthermore, different types of objects exhibit different responses. For instance, linear elastic objects undergo linear deformation, whereas viscoelastic objects exhibit partial resistance to deformations.

For example, to model linear elastic behaviors, the Cauchy strain tensor $\boldsymbol{\epsilon}_C$ given in (2.6) is employed, but its validity is limited to small deformations. Nonetheless, it offers the advantage of fast computation, making it suitable for interactive applications. On the

other hand, the Green strain tensor $\boldsymbol{\epsilon}_G$, given in (2.7), enables the modeling of nonlinear behaviors but comes with a higher computational cost. Consequently, it is less prevalent in real-time implementations.

$$\boldsymbol{\epsilon}_C = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \in \mathbb{R}^{3\text{x}3} \tag{2.6}$$

$$\boldsymbol{\epsilon}_G = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T + \nabla \boldsymbol{u}^T \nabla \boldsymbol{u}) \in \mathbb{R}^{3\text{x}3} \tag{2.7}$$

with

$$\nabla \boldsymbol{u} = \begin{bmatrix} \partial u_x/\partial x & \partial u_x/\partial y & \partial u_x/\partial z \\ \partial u_y/\partial x & \partial u_y/\partial y & \partial u_y/\partial z \\ \partial u_z/\partial x & \partial u_z/\partial y & \partial u_z/\partial z \end{bmatrix} \tag{2.8}$$

Regarding the stress tensor, a common approach for linear objects is to apply Hooke's law, which states that stress is linearly proportional to strain. Mathematically, the relation can be expressed as follows:

$$\boldsymbol{\sigma} = \boldsymbol{E}\boldsymbol{\epsilon} \tag{2.9}$$

with $\boldsymbol{E}$ a tensor that depends on the material properties of the object, such as Young's modulus E and Poisson's ratio ν. Young's Modulus, also referred to as the modulus of elasticity, serves as a measure of a material resistance to deformation when subjected to an external force. It quantifies the relationship between stress and strain within the material, particularly when it experiences an axial deformation. Regarding Poisson's ratio, it expresses the ratio of transverse (lateral) strain to axial (longitudinal) strain when a material encounters uniaxial stress along one direction. In practical terms, Poisson's ratio characterizes how a material changes in dimensions, specifically in width or thickness, when subjected to stretching or compression. This dimensionless value is typically within the range of -1 to 0.5.

The dynamic behavior of the deformation using Newton's law is described as:

$$\boldsymbol{M}\ddot{\boldsymbol{u}} = \boldsymbol{f}(\dot{\boldsymbol{u}}, \boldsymbol{u}). \tag{2.10}$$

where:

- $\boldsymbol{M}$ is the mass matrix of the object;
- $(\dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}})$ are respectively the first and second derivative of $\boldsymbol{u}$.

- $\boldsymbol{f}$ represents both the internal and external forces acting on the object, often formulated as a set of partial differential equations.

By integrating the previous equation, the positions of the model points can be updated and used for simulating deformable objects. Some commonly integration methods include semi-implicit integration, implicit integration, Verlet integration, and Runge-Kutta integration. For a comprehensive review of numerical methods used, readers can refer to the work presented in [Hauth et al., 2003].

2.2.1.1 Conclusion

In conclusion, continuum mechanics serves as a fundamental framework for modeling the behavior of deformable materials, offering valuable insights into soft object deformation, stress, and strain. However, its mathematical complexity often requires the aid of numerical estimation methods to solve real-world problems efficiently. In the following sections, we explore various numerical methods and their crucial role in solving complex problems in continuum mechanics.

2.2.2 Particle-Based Models

Particle-based models are known for their computational efficiency and intuitive nature, however, they lack achieving high levels of physical accuracy. In particle-based methods, the material is represented as a collection of individual particles, each with properties like mass, position, velocity, and possibly other material-specific attributes.

2.2.2.1 Particle Systems

The particles, referred to as model points, initially exist in an equilibrium state that corresponds to their initial positions. When an external force is applied, the object undergoes deformation, resulting in the particles moving to new coordinates. The movement of the particles adheres to Newton's second law (2.10) and is governed by a time integration scheme with considerations for lifespan and collision detection.

Particles are commonly employed for modeling objects like clouds or liquids. However, there are also particle frameworks that utilize dynamically coupled particles to simulate solids [Tonnesen, 1998].

The advantage of particle systems lies in their simplicity, enabling the simulation of complex scenes with a large number of particles. Nonetheless, a limitation of particle

systems is the lack of an explicit surface definition. This drawback can pose challenges in applications such as tracking the restoration of elastic objects to their original form after a deformation was applied by a robot during a manipulation task.

2.2.2.2 Mass-Spring Systems

Another possible particle-based model is the mass-spring model (MSM), which approximates the behavior of a deformable object by the dynamics of a set of masses (points), linked by mass-less springs with dampers as connections to create a volumetric mesh.

Similar to particle systems, the movement of particles is simulated by employing Newton second law of motion. Moreover, a mass point P_i is linked to finite neighbors P_j with springs of rest length l_{ij}^0 and stiffness $\mathbf{K}_{ij} = \text{diag}(k_x, k_y, k_z)$, where (k_x, k_y, k_z) represent respectively the spring stiffness along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Then (2.10) can be expressed for each point as follows:

$$m_i \ddot{\boldsymbol{x}}_i = \boldsymbol{f_s}_i + \boldsymbol{f_D}_i + \boldsymbol{f_e}_i \tag{2.11}$$

where:

- m_i is the mass of any point P_i in a model with N particles, where $i \in \mathcal{N} = [1, ..., N]$;
- $(\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i = \boldsymbol{v}_i, \ddot{\boldsymbol{x}}_i = \mathbf{a}_i)$ are respectively the 3D coordinates of P_i, its velocity and its acceleration;
- $\boldsymbol{f_s}_i$ is the 3D force vector acting on P_i due to springs with stiffness $\mathbf{K}_{ij}$ connecting P_i to its neighbors, P_j, $\forall j \in \boldsymbol{\nu}_i \subset \mathcal{N}$, given by:

$$\begin{aligned} \boldsymbol{f_s}_i &= \sum_{j \in \boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij} = \sum_{j \in \boldsymbol{\nu}_i} \mathbf{K}_{ij} (\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - l_{ij}^0) \frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|} \\ &= \sum_{j \in \boldsymbol{\nu}_i} \alpha_{ij} \mathbf{K}_{ij} (\boldsymbol{x}_j - \boldsymbol{x}_i) \end{aligned} \tag{2.12}$$

- $\boldsymbol{f_D}_i = -D_v \boldsymbol{v}_i$ is the 3D force vector acting on P_i due to the damping D_v.
- $\boldsymbol{f_e}_i$ combines external forces, including gravity.

MSMs offer computational efficiency and intuitive representation, and are used for example for predicting and tracking object states during robotic manipulation, [Petit, Lippiello, Fontanelli, et al., 2017; Schulman, Lee, et al., 2013].

However, a challenge arises in tuning the spring stiffnesses and the damping values of the MSM to achieve the object desired deformation behavior in accordance with its material properties. To address this issue, some approaches proposed learning algorithms [Arriola-Rios & Wyatt, 2017; Bianchi et al., 2004; Morris & Salisbury, 2008]. On the other side, another approach derived an analytical expression for the stiffness of each tetrahedron constituting the mesh, which is proportional to the volume of the tetrahedron and the Young's modulus, [Lloyd et al., 2007]. Another limitation of MSMs is their inability to directly simulate volumetric effects, which prompted to introduce additional energy formulations to account for volume conservation [Teschner et al., 2004]. The orientation of springs also affects the behavior of a MSM, leading to introduce virtual springs to compensate for this effect [Bourguignon & Cani, 2000]. Furthermore, an innovative approach proposed incorporating extra elastic forces into the traditional MSM to accommodate complex mechanical behaviors like viscoelasticity, non-linearity, and incompressibility, [Xu et al., 2018].

2.2.2.3 Conclusion

In conclusion, particle systems offer a valuable tool for approximating continuum mechanics, providing a delicate balance between accuracy and computation time [Etzmuss et al., 2003]. While this section was limited on presenting particle systems and the MSM, it is important to note that various alternative particle-based approaches exist. For instance, there exists a method that uses a recurrent neural network to control MSM dynamics, effectively representing positions, velocities, and accelerations of mass points and springs within a mesh structure [Nurnberger et al., 1998]. Another alternative involves employing a convolutional neural network (CNN) to simulate mechanical loads, opting for the Poisson equation instead of a MSM [J. Zhang et al., 2019].

Next, we present Position-based dynamics, which is an approximation method in continuum mechanics that simulates physical behavior by iteratively enforcing constraints on the positions of discrete elements to achieve dynamic simulations of deformable materials.

2.2.3 Position-Based Dynamics

Unlike particle systems and MSMs, which rely on force-based approaches where model point velocities and positions are updated using a time integration scheme and given forces, position-based dynamics (PBD) models take a different approach. For a detailed

explanation of PBD, readers can refer to [Bender et al., 2017].

In PBD, the positions of particles are updated iteratively by satisfying a set of geometric constraints. These constraints can represent various physical properties and behaviors such as distance preservation, collision avoidance, bending, stretching, or even custom constraints specific to the simulation. Let consider a generic constraint that can be applied to any type of geometric constraint. The constraint equation can be written as:

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N) \approx 0 \tag{2.13}$$

where C represents the constraint function. To compute the updated positions that satisfy the constraint, an iterative approach known as Gauss-Seidel projection can be used. It adjusts the positions of particles by calculating correction vectors based on the constraint function. At each iteration, the correction vector, $\delta\boldsymbol{x}_i$, for each particle i involved in the constraint is computed as:

$$\delta\boldsymbol{x}_i = -\lambda \frac{\boldsymbol{\nabla} C_i}{\boldsymbol{\nabla} C_i^T \boldsymbol{\nabla} C_i} C_i(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N), \tag{2.14}$$

where λ is a stiffness parameter that controls the convergence rate, $\boldsymbol{\nabla} C_i$ represents the gradient of the constraint function with respect to the position of particle i, and $C_i(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)$ represents the value of the constraint function for particle i.

To update the positions of the particles, the correction vector is applied as follows:

$$\boldsymbol{x}_i = \boldsymbol{x}_i + \delta\boldsymbol{x}_i \tag{2.15}$$

By iteratively applying these constraint equations and updating the positions of particles, PBD ensures that the geometric constraints are satisfied while simulating the desired physical behavior.

PBD methods offer several notable advantages. Firstly, they are characterized by their simplicity and computational efficiency. PBD methods do not require a mesh model, resulting in improved memory usage and computational speed. Moreover, their particle-based parallel nature allows for scalability, enabling simulations with a large number of particles. PBD has found extensive application across various interactive graphical applications [Macklin et al., 2014; Tian et al., 2013]. Its utilization has been particularly prominent in modeling the deformation of human body parts [Romeo et al., 2020; B. Zhu et al., 2008] as well as in robotic manipulation tasks [Caccamo et al., 2016; Güler et al.,

2017].

However, PBD has certain limitations. One limitation is that it does not explicitly represent the surface of the deformable object, which can be a drawback when it comes to visualizing or interacting with the object. Additionally, PBD heavily relies on geometric constraints and may require additional techniques or models to handle complex material behavior. On the other hand, MSMs offer greater control over material properties by adjusting the parameters of the springs. By using different types of springs, such as linear or nonlinear, a wide range of materials with varying stiffness, elasticity, and damping properties can be simulated, which allows for more fine-grained control over the behavior of the deformable object.

In the next section, we will explore further numerical techniques, distinct from particle systems and PBD, which are mesh-based techniques and are employed to approximate continuum mechanics in simulations.

2.2.4 Mesh-based Models

In contrast to the particle systems discussed in previous sections, this section delves into mesh-based models as an alternative approach to approximate continuum mechanics. Mesh-based models provide mathematical frameworks for characterizing material behavior, offering a distinct perspective on the analysis of deformable materials and structures.

2.2.4.1 Finite Element Method

The finite element method (FEM) is an effective technique used to approximate the complex physical behavior exhibited by deformable objects. By dividing the deformable body into smaller and simpler finite elements, connected through nodes forming a mesh, FEM allows for accurate representation of the object in response to external forces. Unlike particle-based approaches, FEM operates on node displacements rather than particles. The deformation of a mesh containing N points is determined by computing the displacement vector field $\boldsymbol{u}$, which is obtained through solving (2.16):

$$\boldsymbol{Ma} + \boldsymbol{Dv} + \boldsymbol{Ku} = \boldsymbol{f_e} \tag{2.16}$$

where $\boldsymbol{M} \in \mathbb{R}^{3N\text{x}3N}$, $\boldsymbol{D} \in \mathbb{R}^{3N\text{x}3N}$, $\boldsymbol{K} \in \mathbb{R}^{3N\text{x}3N}$ and $\boldsymbol{f_e} \in \mathbb{R}^{3N}$ are respectively the mass matrix, damping matrix, stiffness matrix and external forces vector in a 3D dimension.

At every time step t, the computation of the matrix $\boldsymbol{K}$ relies on material characteristics like Young's modulus $\boldsymbol{E}$, Poisson's ratio ν, and nodal displacement $\boldsymbol{u}$.

The FEM has seen widespread use in robotics due to its capacity to generate realistic simulations and accurately represent complex deformations. It has been successfully applied in soft object tracking [Petit et al., 2018; Sengupta et al., 2019] and planning manipulation [Frank et al., 2014]. Nonetheless, a limitation of the FEM arises from the necessity to recalculate the dense matrix $\boldsymbol{K}$ at each time step, resulting in significant computational overhead. To address this concern, researchers have adopted an alternative approach wherein the matrix $\boldsymbol{K}$ remains constant. However, this method restricts the model capability to simulate solely small deformations. Alternatively, approaches such as co-rotational FEM [Müller & Gross, 2004] can be employed to reduce computational complexity while maintaining accuracy.

2.2.4.2 Boundary Element Method

The deformation of a surface utilizing the boundary element method (BEM) involves solving the equation (2.16) over the surface, as opposed to the volume-based approach employed by the FEM [Y.-M. Tang et al., 2006]. The surface is divided into N non-overlapping mesh elements, with node coordinates representing the centroids of these elements. In contrast to FEM, BEM offers notable speed improvements by utilizing a reduced number of nodes and elements. However, it is worth noting that the BEM is applicable only to objects with a composition of homogeneous materials inside.

Notably, BEM has found successful applications in real-time volumetric model simulation [James & Pai, 1999], and showed enhanced tracking accuracy in scenarios involving occlusions [Greminger & Nelson, 2008].

2.2.4.3 Modal Analysis

Modal analysis can be utilized to improve the computational efficiency of the FEM by reducing the number of degrees of freedom needed for analysis. The computational burden associated with the FEM arises from the calculation of motion using large matrices $\boldsymbol{M}$, $\boldsymbol{K}$, and $\boldsymbol{D}$ in (2.16). By incorporating modal analysis into the FEM workflow, the computational burden can be alleviated while still capturing the essential dynamic characteristics of the system [Pentland & Williams, 1989]. Here is a breakdown of the mathematical approach for enhancing FEM using modal analysis:

1. Discretization of the system: The continuous system is divided into a finite number of elements and nodes, where each element is defined by its geometry, material properties, and connectivity with neighboring elements.

2. Formulation of the equations of motion: They are derived following the same approach as in the FEM and shown in (2.16).

3. Modal analysis computation: Solve the eigenvalue problem $\boldsymbol{K}\mathbf{v} = \lambda\boldsymbol{M}\mathbf{v}$ to obtain the eigenvalues (λ) and corresponding eigenvectors ($\mathbf{v}$) of the system. The eigenvectors represent the mode shapes of the system, while the eigenvalues determine the natural frequencies and damping ratios.

4. Selection of reduced modes: Choose a subset of modes that significantly contribute to the system response based on criteria like mode participation factors, energy contributions, or engineering requirements. For example, by examining the eigenvalues calculated before, it becomes possible to remove the higher-frequency modes, which allows for the selective update of the most influential modes.

5. Projection of equations onto reduced mode set: Project the equations of motion onto the reduced mode set using the selected mode shapes.

6. Solution of the reduced system: Solve the reduced system of equations considering only the selected modes. This considerably reduces the computational workload as the number of degrees of freedom is reduced to the number of selected modes.

7. Computation of full response: Reconstruct the full response of the system by combining the contributions from the selected modes.

In modal analysis, assuming a constant value for K can enhance computational efficiency. However, this assumption holds true only for simulating small linear deformations and introduces errors when dynamically simulating large deformations. Furthermore, modal analysis offers a robust tool for effectively simulating the characteristics of deformable objects [Basdogan, 2001; Hauser et al., 2003].

2.2.4.4 As-Rigid-As-Possible

The As-Rigid-As-Possible (ARAP) modeling approach is utilized to approximate the behavior of deformable objects by assuming a high level of rigidity. The main objective

of ARAP modeling is to preserve the object shape and rigidity while allowing for local deformations [Sorkine & Alexa, 2007]. This is achieved by formulating an energy function that enforces the rigidity constraint and encourages deformations that closely resemble rigid motion.

The energy function typically consists of a rigidity term and a regularization term. The rigidity term penalizes non-rigid deformations by minimizing the differences in rotations and translations between neighboring vertices. The regularization term promotes smoothness and regularity in the deformation. It penalizes high curvature and encourages a smooth deformation. By minimizing the combined energy function, the ARAP modeling achieves a balance between preserving shape and rigidity and allows for localized deformations, providing a valuable tool for modeling deformable objects.

ARAP has gained significant popularity across diverse applications, such as 3D mesh animation [Levi & Gotsman, 2014], regularization prior for tracking nonrigid scenes [Dou et al., 2017], deformable object tracking during robotic manipulation [Han et al., 2018], and shape servoing of soft objects [Shetab-Bushehri et al., 2022].

2.2.4.5 Conclusion

In summary, the selection of an appropriate modeling approach within the domain of mesh-based models necessitates a careful consideration of the trade-off between precision and computational efficiency. While FEM offers remarkable precision at the cost of increased computational demands, Modal Analysis strikes a balance between accuracy and efficiency, particularly for linear dynamic analysis. On the other hand, the ARAP method excels in maintaining rigidity while prioritizing shape preservation, which is also capable of simulating deformation with less accuracy than FEM. The ultimate choice should be made in accordance with the specific application, the computational resources available, and the requisite level of accuracy for the given task.

2.2.5 Conclusion

In this section, we presented the basics of continuum mechanics and different approaches for approximating continuum mechanics. The choice between these modeling approaches for deformable objects depends on the specific requirements of the application.

Particle systems such as MSM offer simplicity and real-time interactivity, making them a valuable choice for applications where responsiveness and ease of implementation are

crucial. By representing objects as interconnected masses and springs, MSM captures the global deformations and overall structural characteristics of deformable materials efficiently.

On the other hand, PBD presents an intermediate representation between particle systems and mesh-based models, offering a simplified yet efficient approach to enforcing constraints, handling collisions, and simulating deformations in real-time. PBD is well-suited for applications that require a balance between responsiveness and accuracy.

Finally, for high-fidelity simulations with complex material behavior, mesh-based models may be more appropriate but may come at a higher computational cost. For example, by dividing objects into smaller elements, the FEM offers a detailed and accurate representation of deformable objects, making it essential in applications where realism and fidelity are paramount.

In the next section, we will present different approaches for tracking deformable objects and discuss parameter estimation for the physics-based models.

2.3 Deformable object tracking

The initial shape of the object is represented using a topological model before any deformation is applied to it. Then, deformable object tracking is employed to adapt the object shape online during its deformation by continuously tracking it. The process of deformable object tracking can be approached using either a physics-based model or a model-free method. Incorporating an explicit physics-based model can significantly enhance the tracking process by enabling reliable computation and prediction of internal forces experienced by the object. These forces are often not directly measurable by visual or force sensors. Conversely, model-free approaches are better suited for reconstructing soft object models rather than tracking them. Since topological model reconstruction falls outsite the scope of this thesis, we will focus hereafter mainly on the most relevant methods for model-free and model-based visual tracking of deformable objects.

2.3.1 Model-free

Model-Free approaches do not require a predefined model of the object being tracked. Instead, they directly track the object within the camera frames without any prior knowledge about object shape or deformation properties.

2.3.1.1 Linear deformable object tracking by CPD

A framework that encompasses state estimation, task planning, and trajectory planning, all relying on the coherent point drift (CPD) [Myronenko & Song, 2010], was introduced in [T. Tang et al., 2018]. CPD is a registration method that facilitates the non-rigid mapping of one point set to another (see Figure 2.2). We particularly focus on the state estimation proposed in their work, which is applied to a deformable linear object (DLO) using stereo cameras and CPD. A real-time observer was proposed to estimate the states of deformable object by using its corresponding point cloud. Considering that the DLO is often occluded by robot arms or self-occluded during manipulation, the state estimator is designed with robust occlusion handling capabilities. The state estimation process involves acquiring the position of each node on the object by registering the previous estimation results with the new point cloud measurement. As a result, the object state can be robustly estimated in real-time, effectively handling noise, outliers, and occlusions.

The experimental results presented show that the authors approach can successfully and efficiently track a DLO (corresponding in practice to a rope) in real-time, even when dealing with noisy and occasionally occluded point clouds caused by obstacles.

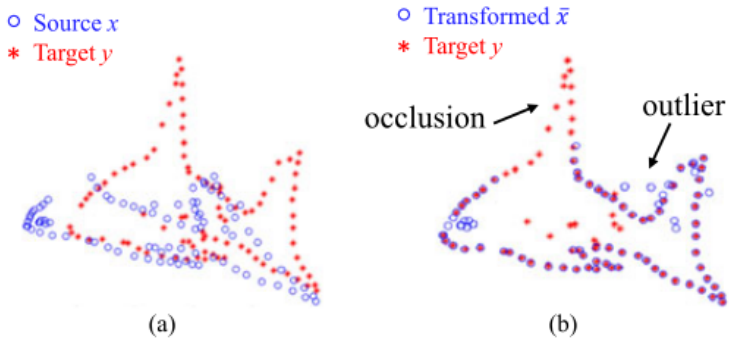


Figure 2.2: Example of CPD registration, from [T. Tang et al., 2018].

2.3.1.2 Occlusion-robust deformable object tracking by regularizing the CPD output

Another model-free occlusion-robust RGB-D sequence tracking framework also based on CPD was proposed in [Chi & Berenson, 2019]. In contrast to the previous approach, the authors stated that CPD only enforces motion coherence between consecutive RGB-D frames, and therefore, there is a possibility of accumulating errors between the current

tracking state shape and the true state. As a result, the topology of the tracking result may deviate from the original model, even though the tracking result remains statistically correct when viewed as a point set registration problem. To address this challenge, the authors propose the use of a regularization term based on locally linear embedding (LLE), wherein LLE performs non-linear dimensional reduction while preserving the local neighborhood structure. This regularization term ensures topological consistency with respect to the original model.

Several experiments were presented, involving the tracking of rope and 2D cloth, to demonstrate the improved robustness of the approach against occlusion. These experiments included both simulation and real-world data.

2.3.1.3 Conclusion

The presented model-free approaches enable the state estimation of 1D and 2D deformable objects by aligning the last step state estimation with the current point cloud measurement, without requiring a physics-based model. Furthermore, these approaches have demonstrated their effectiveness and robustness against object occlusion.

2.3.2 Model-based

Model-based approaches rely on predefined models to describe the object topology, its physics-behavior and its initial undeformed state. Then, they use registration techniques to estimate the update state of the object and optimization techniques to estimate parameters that define an object deformation.

2.3.2.1 Deformable object tracking for model registration and parameter tuning

An approach focusing on tracking and modeling deformation in soft objects using a data-driven approach was presented by [Wang et al., 2015]. Their method involves an iterative framework with two main components: physics-based deformation tracking and an optimization of deformation parameters. The soft objects are modeled using the co-rotational linear Finite Element Method. The optimized deformation model enhances the accuracy of tracking results, leading to improved deformation parameter estimation in subsequent iterations. Notably, their proposed optimization considers not only material elasticity parameters and damping coefficients but also incorporates a reference shape. It refers to the

object geometry subject to zero external forces and can be different from the static shape, which corresponds to the equilibrium geometry under gravity. For a visual representation of the complete pipeline, refer to Figure 2.3.

The process begins with scanning and reconstructing the static shape (Figure 2.3(a)). To capture the dynamic motion, three depth cameras from different viewpoints are utilized, resulting in a time sequence of point clouds (Figure 2.3(b)). Employing a physics-based probabilistic approach and using the static shape as a template, the dynamic point cloud sequence is tracked (Figure 2.3(c)). The result is represented as 3D coordinates of a tetrahedral mesh, closely enclosing the surface template of the static shape. Subsequently, the parameter estimation component optimizes the elastic material parameters, damping coefficients, and reference shape (Figure 2.3(d-e)).

Experimental results validated the ability to track soft objects while simultaneously recovering the reference shape and estimating the deformation parameters. However, this approach necessitates a high accurate initial mesh of the object.

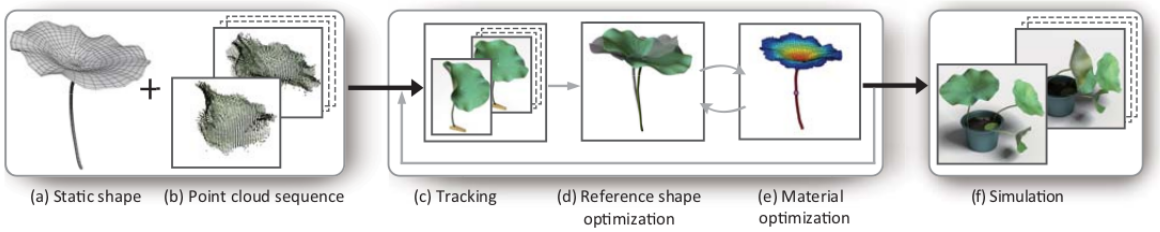


Figure 2.3: The framework of a soft object tracking and estimation of its model parameters, from [Wang et al., 2015].

2.3.2.2 Real-time 3D deformable object tracking using RGB-D camera

Another model-based tracking approach using RGB-D camera, which involves a real-time tracking method designed for 3D soft objects that experiencing significant deformations, including elastic deformations and rigid motions was proposed by [Petit et al., 2015]. The approach considers point cloud data obtained from an RGB-D sensor and utilizes the Finite Element Method for modeling the soft object. It incorporates a consistent mesh representation and takes into account known material properties such as the Young modulus and Poisson ratio. The process involves initially registering the segmented point cloud in a rigid manner, followed by non-rigid fitting of the mesh. Geometrical point-to-point correspondences are used to calculate external forces to be applied on the mesh. The entire workflow is illustrated in Figure 2.4.

The object displacement can be decomposed into a rigid transformation and a pure deformation, with rigid translation and rotation transformations being associated with the rigid transformation. To achieve this, a rigid Iterative Closest Point (ICP) process is applied to estimate a rigid transformation from the resulting segmented point cloud to the mesh. Subsequently, the latter is rigidly aligned to the point cloud data. Then, a deformable registration problem is addressed to fit this point cloud data to the rigidly aligned tetrahedral mesh. This deformable registration involves determining external forces exerted by the point cloud on the mesh and integrating these forces, along with the internal forces computed using the physical model. The computation of external forces relies on geometrical point-to-point nearest neighbor correspondences by employing K-d tree searches from the segmented point cloud to the visible surface of the rigidly aligned mesh and vice versa. Processing both sets of correspondences is crucial because relying solely on a single match from one set to another might lead to inconsistent matches.

Simulation and experimental results confirm that the approach proposed by the authors shows promise as a real-time tracking method capable of effectively managing diverse deformations and motions by combining the FEM, an efficient segmentation method, and conventional point cloud registration techniques.

An extension of the previous work was developed in [Petit, Lippiello, Fontanelli, et al., 2017]. This study involves testing various physical models, such as MSM, FEM, and co-rotational FEM, and comparing the tracking accuracy between these models. Additionally, the researchers applied their method to track the deformation of a pizza in a challenging scenario, where the pizza was manipulated by a humanoid pizza chef robot through stretching and tossing. Another extension was proposed in [Petit et al., 2018], which enables the registration of multiple objects within a single scene. This was not possible before, and it is achieved by effectively handling collision detection model between the different objects of the scene, which are consistently simulated using the co-rotational FEM.

2.3.2.3 Real-time 3D soft tissue tracking in 3D ultrasound images

Another model-based tracking approach to achieve real-time tracking of deformable structures (targets) in 3D ultrasound sequences was introduced by [Royer et al., 2017]. This method involves obtaining the target rigid and non-rigid displacements by minimizing a visual error criteria relying on dense ultrasound visual observation and physics-based simulation. The approach integrates a physics-based model based on a MSM and a 3D object

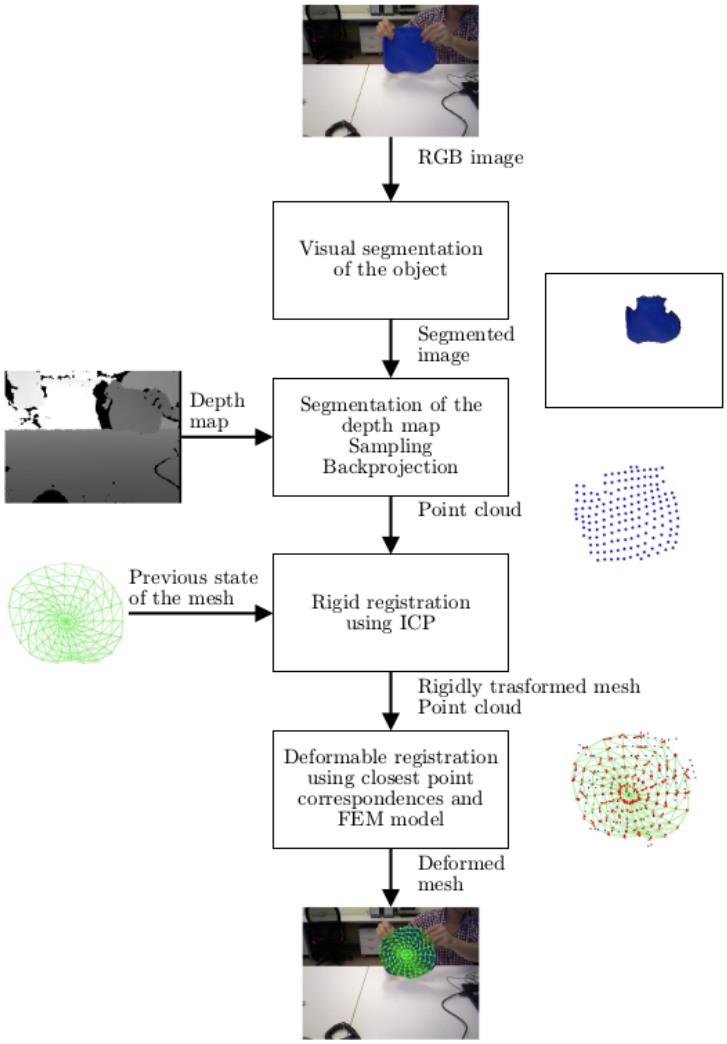


Figure 2.4: The framework of a soft object tracking based on FEM, from [Petit et al., 2015].

mesh to estimate the target motions across consecutive 3D images. For this purpose, the mesh vertices displacements are online updated by iteratively summing their displacements resulted from the MSM internal forces with external displacements that minimize a visual error based on dense ultrasound feedback. This method also includes an ultrasound shadow detection process that increases the tracking robustness by a confidence map used as a weighting mechanism in the visual error criterion. The computational flow of the method is summarized in Figure 2.5.

The primary goal of this approach is to iteratively estimate both the external and internal displacements of the mesh. For computing the external displacements, they adopted an intensity-based method, where the cost function is assessed using various dissimilarity functions such as Sum of Squared Differences, Weighted Sum of Squared Differences, Sum of Conditional Variance, and the newly proposed Sum of Confident Conditional Variance. Concerning the estimation of internal displacements, it is accomplished using the MSM.

The results illustrate that this approach, utilizing the dissimilarity measure Sum of Confident Conditional Variance, provides accurate motion estimation and performs well on ultrasound sequences affected by various challenges, such as speckle noise, large shadows, and ultrasound gain variation. However, it should be noted that the proposed method may encounter inaccuracies when the target experiences significant deformation beyond the limits defined by the provided elastic parameters.

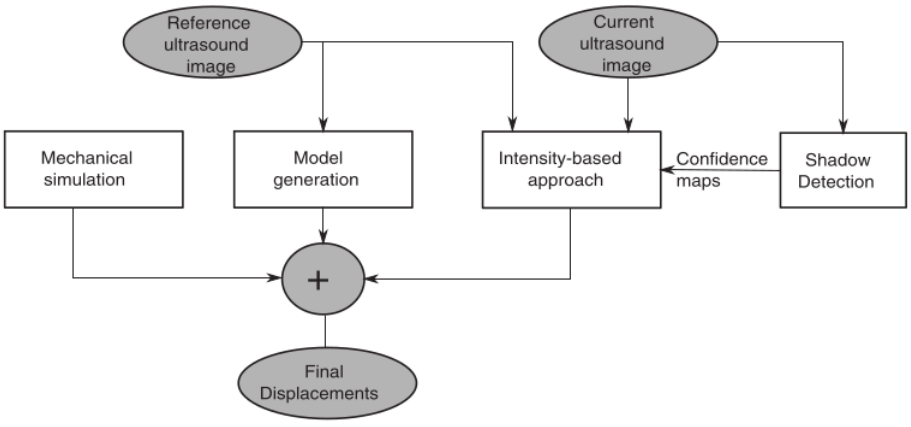


Figure 2.5: The computational flow of the tracking method based on a MSM, from [Royer et al., 2017].

2.3.2.4 Deformable object tracking using coarse physics-based model

Later, [Sengupta et al., 2019] employed a model-based method for tracking deformable objects using a RGB-D camera. They utilized a coarse co-rotational Finite Element Method to represent the physical behavior of the object, and minimize a point-to-plane distance-based geometric error between the point cloud and the mesh by considering virtual forces. The proposed approach does not require an exact physical model of the object or precise properties like Young modulus and Poisson ratio for accurate tracking.

The registration process starts with the same initial step as previous approaches [Petit et al., 2018; Petit, Lippiello, Fontanelli, et al., 2017; Petit et al., 2015], which is rigid registration. For that, two features, namely depth-based geometric error and keypoint-based feature tracking [Trinh et al., 2018], take place. The second step is non-rigid registration. In this part, only a subset of points of interest on the object surface, that are referred to as clusters, are considered to determine the location where a limited number of external forces needed to be applied for deforming the entire mesh. The positions of these clusters correspond to the parts of the mesh that deviate most from the current point cloud. The forces applied on these clusters are then computed to minimize the geometric point-to-plane distance between the RGB-D point cloud and the entire model surface mesh. This is accomplished by establishing a numerical Jacobian that correlates the variation in geometric error with the variation of force applied. Indeed, an Iteratively Reweighted Least Square formulation was employed to minimize the geometric error using this Jacobian.

Simulation and experimental results demonstrate the effectiveness of the authors proposed real-time tracking approach. In contrast to [Petit et al., 2018], the approach achieves accurate tracking of the deforming object even when coarse parameters are considered.

2.3.2.5 Conclusion

The first step of the model-based approach involves generating a 3D tetrahedral mesh model associated with the soft object. Once this model is defined, the approach integrates a physics-based model with the previously described mesh to estimate the target motions across consecutive 3D images. For this purpose, the vertices displacements are calculated by iteratively summing the internal displacements estimated from the mechanical component and the external displacements computed using different methods, such as point-to-point correspondences and an intensity-based approach combined with a shadow

detection process, among others.

2.3.3 Conclusion

In this section, we discussed various approaches to deformable object visual tracking, ranging from model-free methods to model-based techniques. Moreover, deformable object tracking offers diverse applications, one of which is reducing the disparity between real object deformation and its simulated deformation. Reducing this gap enhances the accuracy of the simulated object and allows for fine-tuning of model parameters, as we discuss in more details in the following section.

2.4 Parameter estimation methods

Careful selections of material models and their corresponding parameters is crucial to achieve lifelike soft deformations. The main concept behind the parameter estimation methods is to iteratively adjust the parameters of a simulation system until the deformations obtained in simulation closely match those measured on the real object. In this section, we focus on estimating the elasticity parameters of both MSM and FEM models.

A common preliminary step for all subsequent methods is to use dense 3D reconstruction for object surface mesh generation, comprising N points to represent the undeformed object. Afterward, the model parameters are estimated through an optimization process aimed at identifying the values of the model elasticity parameters $\boldsymbol{p}$ by applying external forces to the object ($\mathbf{f_e}$) and reducing the distance between the model points and a reference model:

$$\hat{\boldsymbol{p}} = \arg\min_{\boldsymbol{p}} \sum_{i=1}^{N} \|\boldsymbol{x}_i(\boldsymbol{p}, \boldsymbol{f_e}) - \overline{\boldsymbol{x}}_i\|^2 \tag{2.17}$$

with $\mathcal{N} = [1, 2, \ldots, N]$, and $\boldsymbol{x}_i(\boldsymbol{p}, \boldsymbol{f_e})$ represents the coordinates of model point P_i, which depend on both the model parameters, $\boldsymbol{p}$, and the applied force, $\boldsymbol{f_e}$. In contrast, $\overline{\boldsymbol{x}}_i$ corresponds to the measured point coordinates obtained from the reference object. This reference model that serves as the ground truth can be either an object captured using an RGB-D sensor or a pre-existing, accurately modeled object. In the latter case, $\boldsymbol{x}_i$ belongs to a coarse model.

2.4.1 Estimation of non-linear heterogeneous FEM parameters

A data-driven approach for approximating parameters of a non-linear heterogeneous soft tissue was presented by [Bickel et al., 2009]. The object is modeled using a FEM and in order to capture material non-linearity, the authors defined the parameter values of the mesh-based model through data interpolation in the strain-space. A gradual load $\boldsymbol{f_e}$ is applied to the object using a probe. At each loading step, with knowledge of the applied force, a sample of the stress-strain relationship is estimated, and the material parameters are computed. By considering measured displacements and applied forces, spatially varying material parameters are computed using an updated version of (2.17) which is given by:

$$\hat{\boldsymbol{p}} = arg\min_{\boldsymbol{p}}\{\sum_{i=1}^{N} \|\boldsymbol{x}_i(\boldsymbol{p},\boldsymbol{f}) - \overline{\boldsymbol{x}}_i\|^2 + \gamma\|\boldsymbol{L}\boldsymbol{p}\|^2\} \tag{2.18}$$

The spatial smoothness of parameters is ensured by employing the sparse Laplacian matrix $\boldsymbol{L}$, while the non-linear residual equation presented in (2.18) is iteratively solved using the Levenberg-Marquardt algorithm [Moré, 2006].

The experimental results validated the approach of estimating the object material elasticity using a data acquisition system that comprised force probes and a marker-based trinocular stereo system. Nevertheless, it is important to note that the approach has certain limitations. For instance, the minimization process may encounter local minima, potentially resulting in inaccurate deformation synthesis. Additionally, the method does not account for viscosity and plasticity parameters.

Another method for estimating the Young's modulus of heterogeneous soft tissues modeled by FEM was proposed by [Coevoet et al., 2015]. The primary innovation involved employing an interactive inverse real-time simulation combined with manual registration of a restricted set of points. This approach relies on utilizing a non-linear FEM within a constraint-based framework. By receiving a limited number of registered points from the user, the method performs real-time optimization to adjust FEM parameters and achieve appropriate geometric deformations. The goal is to estimate the Young modulus from an observed deformation, but for accurate estimation, the applied forces on the model need to be known. To address this, the FEM model equations are projected into the constraint space using Schur complement, effectively reducing the optimization space.

The validation procedure involves several steps. Firstly, an arbitrary deformation of a deformable object is created. Next, few relevant points from the model are selected,

and their positions are recorded when equilibrium is achieved. Then, the simulation is reset without the deformation, and the inverse simulation approach is used to register the chosen points. Finally, a comparison is made between the actual values of parameters used in the process and the values estimated through the inverse method. This comparison is utilized to update the parameter estimation.

Simulation results are utilized to validate the proposed approach for estimating the heterogeneous material, consisting of three distinct Young moduli, one per object part. Various forces were applied to these parts of the object, and the approach effectively estimated the three Young moduli with an error of less than 3%.

2.4.2 Estimation of homogeneous FEM elasticity parameters using a robotic manipulator

Another approach, which focuses on determining the elasticity parameters of a FEM, involves the use of a robotic manipulator equipped with a force sensor and was introduced by [Frank et al., 2010]. The authors estimated the object elasticity parameters, the Young' modulus and the Poisson' ratio, by establishing a relationship between applied forces and resulting surface deformations. The robot exerted a sequence of forces on the soft object while simultaneously observing both the surface of the model object and the real object using a depth camera. Next, an appropriate error function was defined (similar to (2.17)), reflecting the differences between the measured and simulated surfaces observed by the robot. To establish correspondences between the simulated and segmented object surfaces, a 3D registration technique based on point-clouds was utilized. Subsequently, they employed a numerically approximated gradient-descent-based error minimization technique, effectively reducing the disparities between the actual and simulated deformations with respect to the elasticity parameters.

Simulation and experiments confirmed the accuracy of estimating appropriate parameters for various soft objects. To reduce occlusions resulting from the manipulator deforming the object, a thin wooden stick was attached to the end-effector. It is important to note that this approach was restricted to a linear, isotropic, and homogeneous deformation model.

2.4.3 Estimation of MSM parameters using a FEM reference model

A different approach estimating the parameters of an isotropic MSM by leveraging known parameters of the same object modeled with FEM was presented by [Natsupakpong & Çavuşoğlu, 2010]. The authors relied on the premise that FEMs derive accurate material properties of the object, which are assumed to be known. On the other hand, MSMs do not directly approximate the physical behavior of the object. Consequently, the proposed approach aimed to approximate a FEM by optimizing the MSM parameters, while minimizing the matrix norm of the error between the MSM and linear FEM stiffness matrices of the same object.

The approach starts by creating a MSM with the same size and geometry as the FEM, where the elements of the MSM are fully connected. The nodal mass values of the elements in the MSM are assigned to match the nodal elements mass in the FEM, approximating its dense mass matrix by a diagonal one. Concerning the stiffness matrix of each element in the MSM, it is defined based on the unknown spring constants. On the other hand, the stiffness matrix for the FEM element is calculated using a linear elastic model and expressed in terms of the known FEM parameters. Due to the assumption of isotropic material, the MSM element only necessitates two parameters to achieve symmetry, whereas the FEM element possesses two independent elastic parameters. To identify the parameters of the MSM element, both models are subjected to identical displacements. Subsequently, the discrepancy in nodal forces between the two models is assessed using the previously mentioned stiffness matrices. To minimize this error (2.17), an optimization process is undertaken, employing an appropriate matrix norm. It is crucial to note that achieving a zero error is not feasible due to the inherent structural differences between the stiffness matrices of the FEM and MSM elements.

Simulation results demonstrate the efficiency of estimating MSM parameters by using a FEM as a reference model, and minimizing the error between their stiffness matrices through an optimization process. The validity of this approach was further confirmed by validating it across four different configurations. Planar object examples employed triangular and quadrilateral meshes, while three-dimensional volumetric object examples utilized tetrahedral and hexahedral meshes. Despite its effectiveness, the approach does have several limitations. The method is not suitable for arbitrarily connected MSMs, and it calculates the stiffness matrix per element. Furthermore, while there is an alternative

of conducting the optimization at the whole object level instead of per element, it is not feasible due to the high computational cost and the potential risk of encountering local minima problems.

2.4.4 Estimation of isotropic co-rotational FEM parameters using a robotic manipulator

Another approach to estimate the Young Modulus and Poisson ratio parameters within a co-rotational FEM was proposed by [Petit, Ficuciello, et al., 2017]. The authors assumed a continuous isotropic material. The object elastic parameters are determined by tracking its deformations using a RGB-D camera, and incorporating the force measurements obtained from a force sensor. This estimation process follows a similar approach to the one proposed in [Frank et al., 2010], involving the minimization of a fitting error between the real deformations obtained from the RGB-D camera and the simulated ones.

The parameter estimation problem is tackled by minimizing the deviation between the simulated deformations and the observed ones (same form as (2.17)). The innovative aspect of this approach lies in its consideration of distance as the summation of both the distance from the simulated model to the real segmented object and vice versa. Addressing the non-linear optimization problem with respect to the model parameters, the evaluation of the objective function becomes expensive, and computing its gradients becomes non-trivial, thereby making gradient-based optimization methods impractical. To overcome this, the authors employed the gradient-free Nelder-Mead method [Nelder & Mead, 1965].

Experimental tests were conducted to assess the efficacy of the parameters estimation technique. During the experimentation with different initial configurations, convergence of the model parameter estimation was achieved, albeit the presence of local minima was observed.

2.4.5 Estimation of coarse FEM parameters

Another approach to track the deformation of soft objects and estimate their elasticity parameters was presented by [Sengupta et al., 2020]. This method involves fusing visual data obtained from an RGB-D camera with interactive FEM simulations of the object. The study primarily focused on estimating the Young's modulus, assuming that the Poisson ratio was known beforehand. To estimate the elasticity parameters at a contact point of the deformable object, the proposed algorithm relies on the tracking results and external

measurements of the deformation forces obtained at the same point.

The elasticity parameter estimation process involves minimizing the error between a tracked object and a simulated object that is deformed by a robot equipped with a force sensor. External forces are applied at a specific point, and their measurements are stored. Once a steady-state in the deformation tracking is achieved, these measurements are transmitted to the estimation algorithm. A cost-function is computed based on the Euclidean distance error between the object points, where the forces were applied and obtained from the camera, and their corresponding model points at the end of the simulation, where the same forces are replicated. The cost-function is minimized using the Levenberg-Marquardt algorithm. The authors proposed calculating a Jacobian matrix of the cost-function to update the elasticity parameters. This matrix is numerically estimated by conducting multiple simulations with varying parameter values.

Experimental results provide evidence of the effectiveness of their method in estimating the elasticity parameters using the applied forces and a coarse 3D mesh of the object.

2.4.6 Conclusion

In this section, we presented various approaches to estimate the elasticity parameters of deformable objects. These objects can be modeled using discrete models such as MSM or mesh-based models like FEM. The primary goal of parameter estimation is to minimize discrepancies between real object deformations and their simulated deformations when working with real data. Additionally, it serves to reduce disparities between an accurate model of the object and a simplified, potentially less computationally one. Furthermore, parameter estimation proves particularly advantageous in model-based deformable object manipulation, especially when dealing with objects of unknown parameters, as will be discussed in the following section.

2.5 Deformable object manipulation

Robotic manipulation of rigid objects has been studied and successfully implemented in several traditional application fields. However, the manipulation of deformable objects is still an open problem since the object response to an external mechanical action cannot be predicted in the same simple manner as for rigid objects. Manipulating a cable, tearing a 2D cloth, controlling the 3D shape of biological tissues are some examples of

challenging soft object manipulations. Achieving precise and controlled shape changes in these objects is important for a wide range of applications, such as surgical and dressing assistance [Shin et al., 2019; Yu et al., 2017], food handling [Lehnert et al., 2017], and the automotive industry [Shah & Shah, 2016], to name a few. The automation of these tasks combines many areas such as control, perception and modeling/learning, where several issues complicate it: i) the material properties of the object (stiffness, mass, and viscosity), ii) the object geometry (dimension and shape), iii) the forces to be applied (localization, direction and magnitude).

Ensuring stable and dexterous manipulation of soft objects without causing damage is an ongoing challenge. To address it, researchers have developed a variety of shape servoing techniques, which can be broadly categorized into two categories: model-based and model-free techniques. Additionally, the object shape can be represented by some points on its surface, which can be defined by their 3D positions or their pixel coordinates corresponding to the projection of their 3D coordinates in the camera frame. Alternatively, the shape can be defined by a combination of selected object surface points or by considering all surface points.

In the following sections, we begin by reviewing model-free approaches that focus on deformable object manipulation. We then proceed to discuss model-based approaches, which rely on a physics-based model corresponding to the deformable object. Model-based approaches incorporate this physics-based model, while model-free approaches rely solely on visual information.

2.5.1 Model-free

This section concerns the model-free approaches. These methods are mainly based on visual servoing and machine learning, and do not require prior information on deformation behavior deduced from physics-based models. In this thesis, we will specifically focus on the approaches based on visual servoing.

2.5.1.1 Vision-based object 2D deformation features positioning

One of the pioneering works in the field of model-free approaches was introduced by [Navarro-Alarcon et al., 2013]. They presented a novel method for controlling robot manipulators interacting with unknown elastic objects based on visual feedback. Their approach involved estimating the deformation Jacobian matrix in real-time using Broy-

den update rule [Broyden, 2000]. This matrix, as the features Jacobian matrix in classical visual servoing, captures the relationship between the visual feedback features and the robot manipulators motions, relying solely on the visual information. Their closed-loop dynamic-state feedback velocity control law, implemented within a passivity-based framework, ensured input-to-state stability in the presence of imperfect estimations and external disturbances.

The authors introduced four distinct types of features presented in Figure 2.6. These include point-based deformation, which explicitly focuses on displacing singular/multiple 2D points. Distance-based deformation regulates the distances between point features. Angle-based deformation focuses on the relative angle of a line of interest constructed from two 2D points. Finally, curvature-based deformation involves measuring and controlling the contour of interest defined from three 2D points. These feature vectors were defined as nonlinear functions of the visual feedback points.

Afterwards, the change in these feature vectors was represented in relation to the robot manipulators using a Jacobian matrix. Utilizing this Jacobian matrix, a control law based on energy-based dynamic-state feedback velocity was derived. The objective of this control law was to convert the deformation behavior into a non-conservative Hamiltonian dynamical system, aiming to achieve the desired deformation by minimizing an energy-like functional. The authors stated that even with imperfect estimations of the deformation Jacobian matrix, the closed-loop system exhibited input-to-state stability and complete dissipativity in the presence of external disturbances.

The experimental results demonstrate the effectiveness of their control method in handling real-time deformation tasks without the need for camera calibration parameters or modeling of the deformation. However, it should be noted that the proposed method has certain limitations. It requires slow motion of the robot manipulators and low-pass filtering of the observation signals. Additionally, the method is applicable only to quasi-static and elastic objects.

Subsequently, two vision-based control methods for shaping soft objects, both of which did not rely on the identification of the deformation model or camera parameters, was introduced in [Navarro-Alarcon et al., 2014]. The first controller enables real-time estimation of the deformation Jacobian matrix. The second approach introduces an adaptive deformation controller and combines a minimal amount of offline information with online adaptation of variable parameters. Prior to conducting the experiments, the method requires gathering some test points, which involves non-collinear end-effector displacement

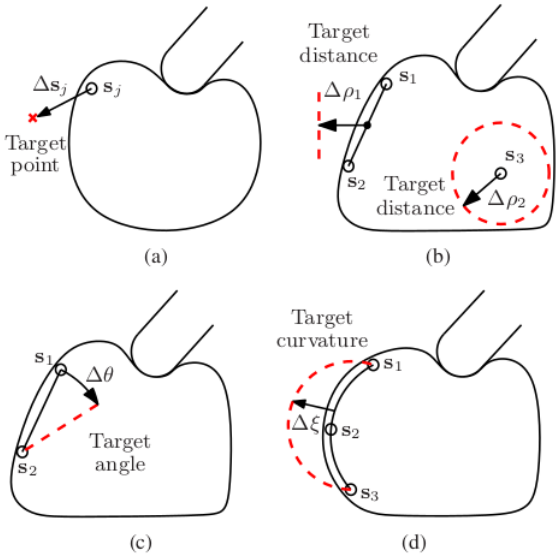


Figure 2.6: Different types of feature-based deformation, from [Navarro-Alarcon et al., 2013]: (a), (b), (c), and (d) correspond to point-based deformation, distance-based deformation, angle-based deformation, and curvature-based deformation, respectively.

vectors and visual feedback vectors. These vectors are used to construct a regression matrix, which remains constant throughout the online deformation test. This regression matrix is subsequently used to create an estimated projection model for a feature point onto the image plane (see Figure 2.7). The parameter adaptation rule is determined by a combination of two terms. The first term represents the difference between the current position and desired position of the test points. The second term accounts for the error in the estimated projection model (see Figure 2.7).

The effectiveness of both controllers was evaluated by implementing them on a robot manipulator. However, there are still several limitations associated with these proposed controllers. The first approach is sensitive to image noise, but the authors addressed this issue by employing a low-pass filter on the signals. On the other hand, the second control method is less affected by noise, but it may not be suitable for tasks that require a broad range of nonlinear deformations. This method is therefore better suited to the applications of small or localized deformations to the object.

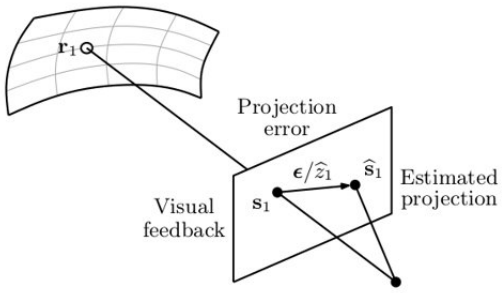


Figure 2.7: Geometric representation of the projection estimation error, from [Navarro-Alarcon et al., 2014].

2.5.1.2 Simultaneous object positioning and shaping

In a subsequent work, a model-free method for automatic 3D shape servoing of soft objects using robotic manipulators was proposed by [Navarro-Alarcon et al., 2016]. The object shape is represented by a deformation feature vector consisting of two distinct components: position and shape components. This approach enables the manipulation of an object to reach a desired position while controlling its relative deformation. The pipeline of this method is depicted in Figure 2.8.

The authors stated that the deformation model can be parameterized linearly using an adaptive vector of unknown deformation parameters. Subsequently, they developed an algorithm that dynamically estimates this vector and iteratively approximates the deformation model, which does not necessarily correspond to the true deformation model. Afterwards, the estimated vector is used to approximate the deformation Jacobian matrix, which is necessary for computing the velocity control input with conventional controllers. Furthermore, the visual feedback consists of two components: a positioning feedback term and a shaping feedback term. The first term involves the location of explicit feature points or the geometric center of multiple feature points, and is used to control the object position. The second term evaluates the internal displacements of the object, allowing for control of relative deformations. It encompasses the compression distance, which measures the relative distance between two feature points. Additionally, a folding angle of the object is defined as the measured angle between two intersecting lines computed using feature points and a normalized curvature that quantifies the roundness of the object. All these shape geometric descriptors are then servoed by the controller to reach desired values.

The method was tested using two manipulators and the proposed controller to ma-

nipulate the shape of a soft object and drive it towards a desired shape. However, this method does have certain limitations. For instance, the method is specifically designed for slow movements of the robots and may not be effective in controlling rapidly changing shapes. Another limitation is the assumption of the object model being purely elastic.

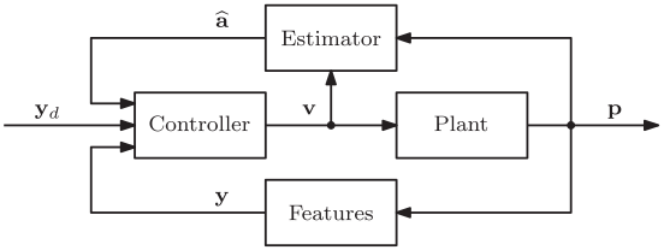


Figure 2.8: The pipeline of a 3D shape servoing proposed in [Navarro-Alarcon et al., 2016].

2.5.1.3 2D contour manipulation

In a later study, a model-free approach for automatically deforming soft objects into desired two-dimensional shapes using robot manipulators was proposed by [Navarro-Alarcon & Liu, 2017]. The shape of the object is represented by its contour, and a compact feedback characterization of deformable object shapes is developed based on a truncated Fourier series. This method differs from previous approaches that typically rely on displacements of feature points to define the servoing task. Instead, it introduced closed-loop deformation tasks by utilizing the truncated Fourier series of 2D image contours as a form of visual feedback. The closed-loop controller is depicted in Figure 2.9.

The object contour can be reconstructed using the shape parameters derived from the Fourier approximation. Therefore, the shape error corresponds to the difference between the Fourier coefficients of the current shape representation and the one corresponding to the desired shape. Regarding the control law, it was designed using a Jacobian-based approach. To estimate the Jacobian matrix and to overcome the challenges associated with the high dimensionality and nonlinear behavior of the object model, the authors proposed an online algorithm to approximate the deformation properties. This approach avoids the requirement of full parametric identification of the object model, as seen in previous approaches. The proposed method employs a local deformation model that is recalibrated at different local regions. As the robot moves into a new configuration, the model is recalibrated using data points collected along the local trajectory. The objective

of the iterative model estimator is to compute a local approximation of the Jacobian matrix. However, it is important to note that since the estimator computes a locally valid linearized model, it cannot capture the entire properties of the object in general.

The approach was validated through both numerical simulations and experimental results. It is important to note that the proposed method is specifically designed to control the 2D image projections of object contours and solely involves translational movements. Additionally, it is worth mentioning that the system run into difficulties if the desired contour is unreachable or if the number of Fourier coefficients is insufficient to accurately describe the object shape. In such cases, the system may converge to a local minimum configuration.

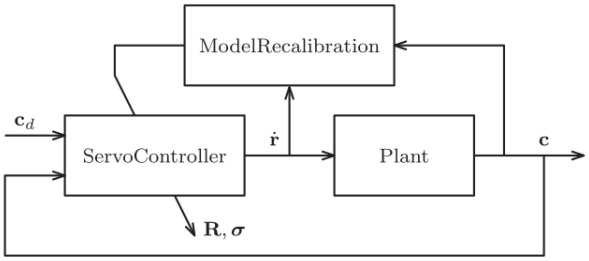


Figure 2.9: The pipeline of a model-free shape servoing using Fourier series represented in [Navarro-Alarcon & Liu, 2017].

Another approach, which focuses on controlling the shape of soft objects, was introduced by [Qi et al., 2021]. In this approach, the object shape is also represented through its contour extracted from a 2D image of the observed object. The object contour was parametrized using a set of 10 parameters. These parameters include the object centroid coordinates (2 parameters), the logarithmic function of the 7 Hu moments [M.-K. Hu, 1962], and the orientation of the contour principal axis in the plane [Zheng et al., 2019]. These parameters are subsequently normalized. Similar to previous methods, this approach establishes a relationship between changes in the object shape parameters and the robot motions through the use of a deformation Jacobian matrix. Subsequently, a sliding mode controller was proposed to automatically reshape the object to a desired configuration while simultaneously estimating the Jacobian matrix. To assess the effectiveness of this approach, experiments were conducted to validate the proposed control and reshape various elastic and plastic objects to achieve desired contours. It is worth noting, however, that this method comes with certain limitations. It is not suitable for purely plastic objects, and the object deformation process must occur slowly.

2.5.1.4 Fourier-based 1D shape control

An alternative method employing Fourier series for model-free shape servoing was proposed by [J. Zhu et al., 2018]. The study focused on manipulating a 1D linear cable, where the shape of the cable was approximated using Fourier series. The manipulation was carried out using two robotic manipulators. The objective of this research was to establish a framework for multi-arm robots to effectively deform a flexible cable into a desired shape through vision-based control. The local deformation model was estimated online using shape parameters, allowing for real-time adaptation. The local deformation model is similar to the one proposed in [Navarro-Alarcon & Liu, 2017], with the distinction being that this method incorporates multiple objects instead of just one, along with the inclusion of rotational action. A small displacement of the robot results in a minor alteration in the cable shape and subsequently affects the feature parameters. Exploiting this observation, the authors were able to linearize the deformation model. With the deformation model, a velocity control law was applied to the robot, enabling the deformation of the cable into the desired shape. The effectiveness of the method was successfully confirmed through validation using two grippers that manipulated different cables to achieve the desired shape.

2.5.1.5 3D positioning of feature points

Another approach for shape servoing of soft objects involving an online estimation of the deformation Jacobian was introduced by [Lagneau et al., 2020a]. The shape of the object is represented through 3D points of interest and the Jacobian estimation employs weighted least-squares minimization within a sliding window. Subsequently, the obtained deformation Jacobian is employed in the control law to manipulate the object shape.

In comparison to alternative model-free approaches, the study conducted by [Lagneau et al., 2020a] employed a limited number of parameters for online estimation of the deformation Jacobian. By contrasting their method with the approach utilizing the Broyden update rule, the authors concluded that their technique exhibits greater resistance to noise. This is achieved by updating each row of the deformation Jacobian matrix based on a confidence criterion derived from the observed deformations. Specifically, in their approach, the Jacobian is updated only for degrees of freedom (DOFs) that meet the confidence criterion. This confidence criterion serves as a filter for noisy measurements when the system approaches the desired deformation. The update of the estimated defor-

mation Jacobian is determined by a user-defined confidence threshold, and the confidence criterion is determined using eigenvalue decomposition to ascertain how the Jacobian will be updated.

Multiple experiments were conducted to verify the capability of controlling the shape of soft objects. In [Lagneau et al., 2020a], they presented various shape-servoing tasks involving 3D objects, where they drived specific points of interest on the object surface to achieve desired positions. The authors demonstrated that their technique remains independent of the quantity of points of interest and external disruptions. The experiments revealed that the proposed method achieved comparable accuracy to state-of-the-art methods, all while relying on a smaller number of parameters to tune. Furthermore, they validated in [Lagneau et al., 2020b] their approach by actively controlling the 3D shape of a wire, using two robotic manipulators to control its two extremities and shape it to reach a desired 3D form.

2.5.1.6 Conclusion

In this section, we presented various model-free approaches that focus on deformable object manipulation. The common idea among these methods is their attempt to estimate the Jacobian matrix, which relates the variations of the visual feedback features to the manipulator motions, purely from visual observation and robot odometry. These approaches do not rely on a physics-based model of the deformable object, and in some cases, they do not require knowledge of the camera calibration used to capture the feedback information. However, these methods can be sensitive to noise since their Jacobian matrix is directly derived from inherently noisy visual data. To address this issue, some approaches employed data filtering techniques before constructing the Jacobian matrix, while others design robust controllers to mitigate measurement noise.

In addition to the mentioned Jacobian-based approaches, there exist numerous learning-based approaches that have been suggested for deriving the deformation model or manipulation strategies directly from data. Here, we will mention a few of them. For instance, [Schulman, Gupta, et al., 2013] used depth images to encode the status of a manipulation task of soft objects and learning from demonstration to allow a robot to execute multi-step manipulation strategies. This technique has been expanded upon using reinforcement learning [Hadfield-Menell et al., 2015] and tangent space mapping [T. Tang et al., 2016]. A deep learning-based end-to-end framework has also been proposed in [Yang et al., 2016]. Furthermore, [Clegg et al., 2017] used reinforcement learning for haptic manipulation. In

this case, the use of a high-quality simulator was crucial for the success of the learning process. Finally, [Z. Hu et al., 2019] employed deep learning to estimate the deformation Jacobian matrix. This matrix was subsequently used to generate a control law aimed at controlling both the position and shape of deformable objects.

2.5.2 Model-based

Model-based deformable object manipulation refers to the process of manipulating objects using their physics-based models. A physics-based model is used to simulate the actual behavior of the object when exposed to external forces. In this section, we review different approaches on soft object shape servoing based on object models.

2.5.2.1 Indirect positioning of object feature points

Some pioneering studies have addressed the indirect positioning of soft objects, where multiple selected object points, denoted feature points, are driven to their desired positions indirectly by manipulating other points of the object, denoted as manipulated points.

One of the first works that proposed a control approach for simultaneously positioning multiple points on deformable textile fabrics was presented by [Wada et al., 1998]. The operation points in Figure 2.10(a) correspond to the manipulated points, while the positioned points refer to the feature points. Their method involved creating a simplified physics-based model using a two-dimensional MSM system with different spring stiffnesses and initial lengths along horizontal, vertical, and diagonal axes, as presented in Figure 2.10(b).

By employing visual sensors and a linearized model in a quasi-static equilibrium state, their control method effectively guided the positioned points to desired locations, even without precise knowledge of the fabric physical properties.

Subsequently, in the work conducted by Kinio and Patriciu [Kinio & Patriciu, 2012], the object was modeled using the FEM and a H∞ controller was employed for the purpose of accomplishing the positioning task. This study involved two manipulated points and two feature points. The authors demonstrated via simulation and experimental results that in this case the H∞ controller presents a more advantageous design option compared to PID (Proportional-Integral-Derivative) controllers.

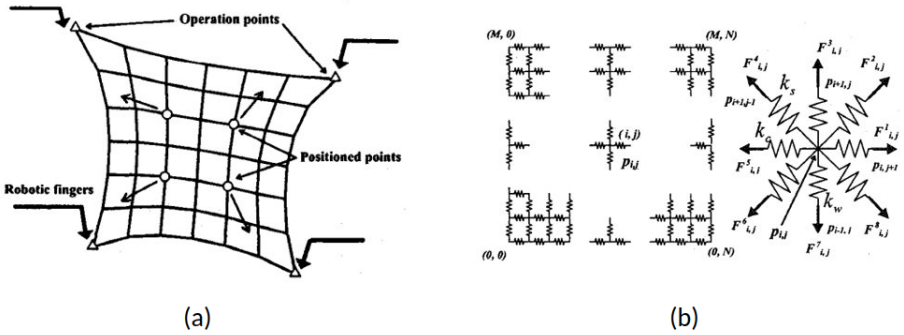


Figure 2.10: Indirect positioning of multiple feature points, from [Wada et al., 1998]: (a) and (b) depict respectively the distribution of mesh points and the spring model of the 2D object.

2.5.2.2 Two-phase strategy for object modeling and shaping in one dimension

A two-phase strategy for shape servoing, which is depicted in Figure 2.11(a), was introduced by [Higashimori et al., 2010]. In the first phase, the object is modeled using a four-element viscoelastic model that included two elements for elasticity and two for viscosity, then its parameters are estimated. The estimation process involved applying a contact force to the object in a step-wise manner, as shown in Figure 2.11(b). The force was increased up to a threshold level to avoid damaging the object, maintained for a specific duration, gradually reduced to zero, and finally, the contact was maintained without applying any force indefinitely. The resulting deformations during non-zero force application represented the elastic part (red plot), while the maintained deformations after unloading represented the plastic part (green plot). In the second phase, the authors actively controlled the plastic deformation response to achieve the desired final deformation. By using the estimated viscoelastic and plastic parameters, they determined the necessary force required to achieve the desired shape of the object.

Experimental validation was conducted to deform a 3D parallelepipedic object in one direction.

2.5.2.3 2D object contour servoing using MSM

[Das & Sarkar, 2011] proposed an approach for manipulating the shape of deformable objects, using a system of manipulators. The initial and final shapes of the object are defined by boundary curves. The object is modeled using a MSM where the connections between nodes are modeled as a Voigt model, as shown in Figure 2.12(a). The Voigt

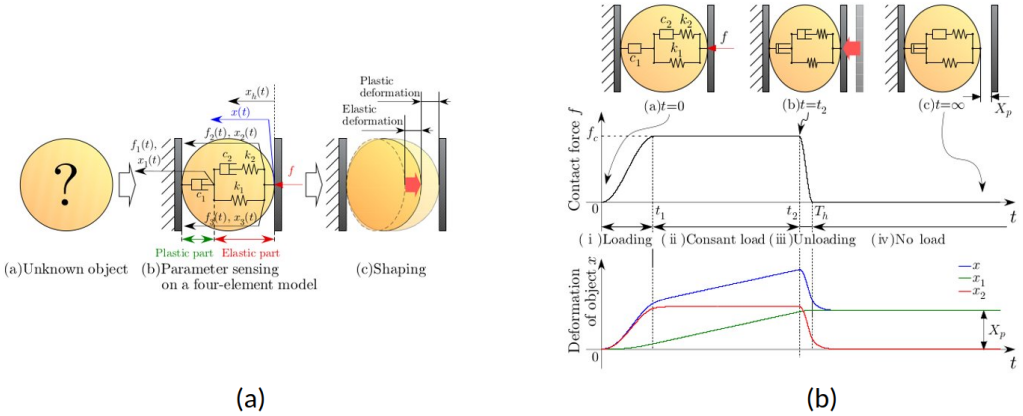


Figure 2.11: 1D shape deformation of an unknown soft object, from [Higashimori et al., 2010]. In (a), the two-phase strategy for shape servoing of an unknown rheological object is presented. In (b), the decomposition of the object deformation response is illustrated, with x_1 representing the elastic response and x_2 representing the plastic response.

model consists of a parallel arrangement of a spring and a damper. To achieve the desired shape change, an optimization-based planner is designed, which minimizes an energy-like criterion to determine the optimal locations of the contact points along the final shape curve. Importantly, each manipulator operates independently without the need for communication between them. Additionally, a robust controller was derived to handle modeling uncertainties.

Two simulation results provided evidence of the effectiveness of the proposed method. The first simulation involved applying the shape controller to deform a circular shape into either an ellipse or a square, using varying numbers of actuation points. The outcomes indicated that a higher number of actuation points resulted in improved system accuracy. In the second simulation, a deformable object was manipulated by a three-finger robotic hand without causing deformation. The robust controller effectively achieved this objective. Figure 2.12(b) illustrates an example of the first simulation task.

2.5.2.4 FEM-based soft robot control

An alternative approach for controlling the shape of soft objects was proposed in the context of soft robots. For example, the robot needs to deform itself to maintain specific configurations and accomplish given tasks. In this context, an approach that includes two controllers (see Figure 2.13) for the manipulation of soft robots, utilizing visual tracking and a simulation-based predictor, was introduced in [Z. Zhang et al., 2017].

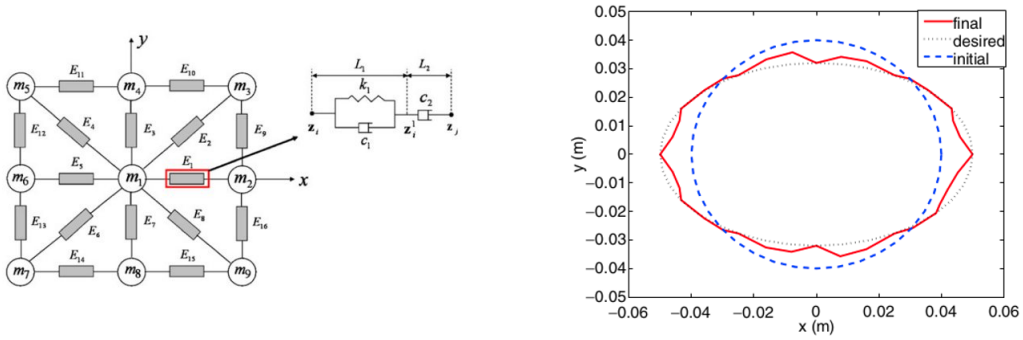


Figure 2.12: 2D shape deformation of a soft object presented in [Das & Sarkar, 2011]. In (a), a model of a 2D object is presented. In (b), the shape transformation from a circle to an ellipse is represented using 12 manipulators. The initial shape, desired shape, and final shape are indicated by blue dashed, black dotted, and red solid lines, respectively.

The first controller, depicted in Figure 2.13(a), is designed to guide the soft robot end effector to a desired position by applying a control law. This controller utilizes a simulation-based predictor that calculates a Jacobian matrix using the kinematic model of the soft robot, where a real-time FEM is employed. The open-loop controller consists of two parts: closed-loop control of the simulation model and open-loop control of the soft robot. The closed-loop control ensures that the Jacobian matrix of the simulation model remains close to that of the real robot, while the open-loop control ensures that the soft robot end effector tracks the desired trajectory. It is worth noting that the first controller is still considered open-loop, even with the presence of closed-loop control, as it does not require feedback from the soft robot. In this approach, both the simulation model and the soft robot utilize the same control input, which is computed based on the contribution of the actuators derived from the simulation model.

The second controller (see Figure 2.13(b)) utilizing infrared cameras and feedback is developed to correct the position of the end effector. This controller is implemented to actuate the simulation model, aligning the end effector of the simulation model with that of the real robot. Consequently, both the simulation model and the soft robot exhibit similar configurations.

To evaluate these methods, the authors conducted experiments on a soft robot actuated by cables.

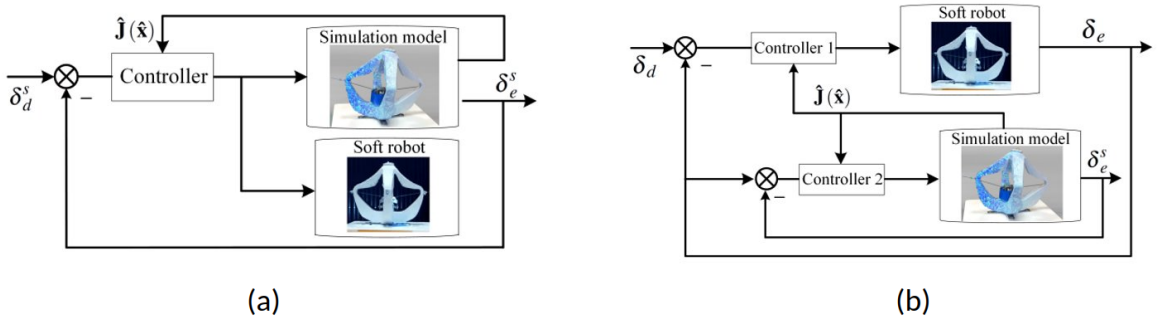


Figure 2.13: The implementation schema of the two controllers proposed in [Z. Zhang et al., 2017]: (a) and (b) correspond to the implementation of the open-loop controller and closed-loop controller, respectively.

2.5.2.5 FEM-based 3D deformable object elasticity parameters estimation and open-loop shape control

Another approach based on FEM to generate open-loop forces on anthropomorphic fingertips and achieve desired displacement on a soft object was proposed by [Ficuciello et al., 2018]. The methodology is depicted in Figure 2.14. Instead of using objects with known parameters, they employed a vision system and a force sensor to estimate the Young's modulus $\boldsymbol{E}$ and Poisson's ratio ν. This estimation relied on the tracking technique introduced by [Petit, Lippiello, Fontanelli, et al., 2017], which minimizes the fitting error between simulated and RGB-D sensor-captured deformations. The authors tackled the nonlinear optimization problem of parameter estimation using the gradient-free Nelder-Mead method [Nelder & Mead, 1965].

Once the model parameters were estimated, the objective is to minimize the deviation between end-effector positions and desired positions. This task was accomplished using an online inverse simulation technique, employing convex optimization, for adjusting the applied forces at the fingertips during contact and achieving the desired deformation.

Additionally, since the deformation was performed in an open-loop fashion, there was an inherent error between the expected and actual deformations on the real object. The authors identified three sources contributing to this error: convex optimization, model parameter approximations, and hardware limitations such as sensor uncertainties. The experimental results presented the deformation achieved by the underactuated anthropomorphic SCHUNK 5-Finger Hand (S5FH) on a cylindrical object.

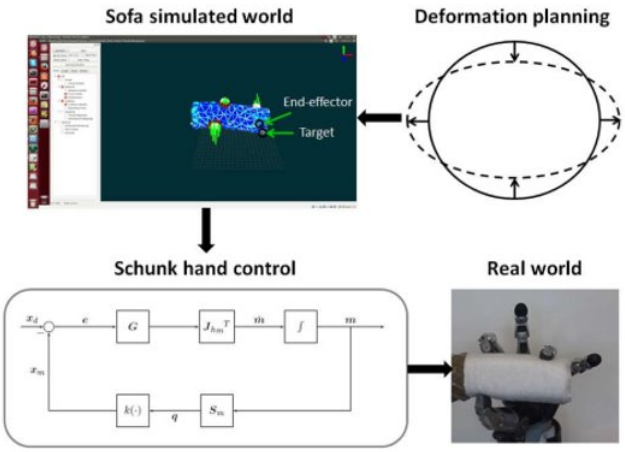


Figure 2.14: The pipeline for an open-loop deformation control using FEM, from [Ficuciello et al., 2018].

2.5.2.6 ARAP-based planar object shape servoing

An alternative method for shape servoing, wherein a geometric mesh was utilized to represent the object surface instead of a physical mesh was introduced in [Shetab-Bushehri et al., 2022]. The authors stated that in numerous practical scenarios of deformable object manipulation, the object tends to naturally maintain local rigidity. Based on this observation, the authors introduced a shape-servoing approach based on the As-Rigid-As-Possible (ARAP) modeling [Sorkine & Alexa, 2007]. The authors further justified their motivation by highlighting that this type of modeling eliminates the need to estimate mechanical parameters of the object. The authors proposed a control scheme that aimed to guide points of a planar object to desired positions through a closed-loop system using the complete object surface points as a feedback measurement. To obtain the object points, the authors therefore tracked the object during the deformation.

The proposed control scheme integrates an estimation of the deformation Jacobian matrix based on the ARAP model. The Jacobian was approximated by establishing a connection between the robot end-effector frame and the ARAP mesh. In this approach, the changes in the features correspond to the variations in the model surface points, while the velocities correspond to the applied motions on the object surface by the robot end-effectors. Then, to obtain an initial approximation of the Jacobian matrix, the undeformed shape of the object was taken as a starting point, and a perturbation was simulated in one degree of freedom (DOF). This process results in a new stable shape for the object

after the perturbation. Subsequently, a standard forward finite differences method was utilized to estimate the column of the Jacobian associated with the perturbed DOF. By repeating this procedure for all DOFs, an initial approximation of the Jacobian matrix was obtained.

Regarding the shape tracking pipeline, it was developed using the approach introduced in [Aranda et al., 2020]. This pipeline utilizes monocular vision and is built upon the principles of shape-from-template. It relies on the presence of visual texture on the object surface for accurate matching and assumes that the object surface undergoes isometric deformation.

The authors made the assumption that the object shape remains statically stable, achieving quasi-static equilibrium, and responds smoothly to robot motions. They conducted several shape servoing tasks using two Franka Emika robot arms and four different objects made of various materials to validate the effectiveness of their proposed scheme. An example of a shape servoing task accomplished in this work is presented in Figure 2.15. To ensure that the desired shape could be achieved, each experiment started by manually manipulating the two arms holding the object to record the desired shape. Afterward, the deforming object was manually adjusted to its initial shape using the two robot arms before starting the task.

2.5.2.7 Conclusion

In this section, we presented a review of various model-based approaches for manipulating soft objects, along with a related work concerning the deformation of soft robots. The control laws developed to accomplish these tasks depend on the object models employed to approximate the physical behavior of the object. These approaches either assume that the model parameters are known in advance or involve estimating them before the manipulation process.

2.5.3 Conclusion

Tables 2.1 and 2.2 present a summary of different methods presented in this section and used to deform soft objects.

Model-free approaches can be categorized into two distinct groups. In the first category, known as numerically estimated Jacobian-based approaches, some authors argue that there is no need to estimate the object model or its physical parameters. Neverthe-

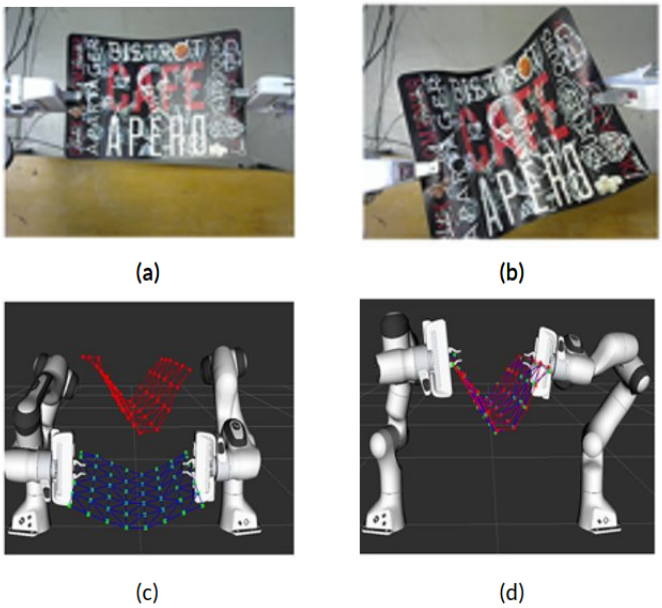


Figure 2.15: Illustration of a shape servoing task using the ARAP model, from [Shetab-Bushehri et al., 2022]: (a) and (b) represent the camera view of the object before and after the deformation task, respectively. (c) and (d) illustrate the object surface points before and after the deformation task, respectively. The blue points correspond to the actual object surface points, and the red ones correspond to their desired positions.

less, other methods within this category attempted to estimate these parameters during the deformation process. These numerically estimated Jacobian-based models primarily consider the quasi-static equilibrium of the object, limiting their applicability to dynamic objects. Furthermore, since these approaches are mainly based on the numerical estimation of a deformation Jacobian from data obtained from sensors, they are particularly sensitive to measurement noise. In the second category, machine learning techniques have garnered attention, but they require large datasets for effective model training.

In contrast to model-free approaches, there are model-based methods that rely on physics-based models. These approaches not only take into account the object geometric shape, often described by a 3D mesh, but also consider its mechanical properties. These models aim to replicate the actual behavior of the object when subjected to external forces. However, it is worth noting that these approaches require precise knowledge of the object model parameters. These parameters have to be known in advance, or accurately identified before the manipulation task.

2.6 Positioning of this thesis in relation to the existing literature

In the preceding sections, we provided an overview of the state-of-the-art in deformable object modeling, tracking, and deformation. Our purpose in presenting these topics was to enquire the reader with recent developments in the literature and to provide a clearer introduction to the objectives of this thesis. In this thesis, we aim to introduce novel physics-based methods for deformable manipulation, utilizing a simple mass-spring model.

Unlike the previously mentioned approaches in Section 2.5, we introduce a novel vision-based and physics-based control law for shaping deformable objects, aiming to employ a simple model that has the advantage of a low computational cost to enable fast shape servoing tasks. The motivation for combining vision-based and physics-based approaches arises from advancements in computer graphics, where various physics-based models were proposed to represent deformable objects, and from computer vision, which aids in tracking such objects, ultimately reducing the gap between the object simulated deformation and its real one. In our approach, the vision component plays a crucial role, where the object is tracked during deformation. If the object model deviates from the actual object deformation, we adjust the object model by applying constraints to it. Regarding the object model, it imitates the physical behavior of the real object, with its associated

Approach	Category	Dimensionality	Feedback information	Hypotheses/Notes
[Wada et al., 1998]	Model-based (MSM)	Planar objects	Object 2D points	Quasi-static assumption
[Higashimori et al., 2010]	Model-based (four-element viscoelastic model)	3D objects	Object length	Deformation in one direction
[Das & Sarkar, 2011]	Model-based (MSM)	Planar objects	Object contour points	
[Kinio & Patriciu, 2012]	Model-based (FEM)	Planar objects	Object 2D points	
[Navarro-Alarcon et al., 2013]	Model-free	3D objects	2D deformation features	Quasi-static assumption
[Navarro-Alarcon et al., 2014]	Model-free	Linear and planar objects	2D deformation features	Quasi-static assumption and small deformations
[Navarro-Alarcon et al., 2016]	Model-free	3D objects	2D position and deformation features	Quasi-static assumption and slow deformations
[Navarro-Alarcon & Liu, 2017]	Model-free	3D objects	Object 2D contour parameterized using Fourier series	Quasi-static assumption
[Z. Zhang et al., 2017]	Model-based (FEM)	Soft robots	Robot configuration	Quasi-static assumption
[Ficuciello et al., 2018]	Model-based (FEM)	3D objects	End-effector positions	Quasi-static assumption and open-loop deformation

Table 2.1: Summary of deformable object manipulation approaches (Part I)

Approach	Category	Dimensionality	Feedback information	Hypotheses/Notes
[J. Zhu et al., 2018]	Model-free	Linear objects	Cable pixel coordinates parameterized using Fourier series	
[Lagneau et al., 2020a]	Model-free	3D objects	Object 3D points	
[Lagneau et al., 2020b]	Model-free	Linear objects	Cable 3D geometric features based on B-splines	Valid for small deformations and adapted for larger ones
[Qi et al., 2021]	Model-free	Linear and 3D objects	Object 2D contour parameterized using image moments	Quasi-static assumption and slow deformations
[Shetab-Bushehri et al., 2022]	Model-based (ARAP)	Planar objects	Object 3D points	Quasi-static assumption and shape local rigidity.

Table 2.2: Summary of deformable object manipulation approaches (Part II)

parameters obtained prior to the deformation process through a few simple steps.

In contrast to approaches that solely rely on models, particularly model-based methods, our approach utilizes a simple mass-spring model and does not require precise knowledge of the exact parameters of the soft object. Compared to model-free approaches, we analytically develop the formulation that provides the variation of the object deformation in function of the motion of one or multiple robotic manipulators using this simple mass-spring model. Moreover, our approach does not rely on the assumption of quasi-static equilibrium that is generally considered in the existing methods of the state-of-the-art. As a result, our developed control law will be analytically derived, taking into consideration the dynamic nature of the object.

Additionally, our approach differs from recent model-free methods in how object feature points and shape are represented thanks to the use of a RGB-D camera. Regarding feature points, instead of relying solely on their 2D projections in the sensor image (pixels), we use their 3D positions. Concerning object shape representation, we avoid describing it solely through the projection of its contour, which heavily depends on the camera pose relative to the object model, or by using specific 3D points from the object surface. Instead, we represent 3D objects using their complete set of surface points, while for 2D objects, we define the shape through their 2D contour points. Then, we represent the object shape using a low-dimensional feature vector. Unlike the use of 2D Fourier descriptors or 2D moments for 3D objects, as proposed in the state-of-the-art approaches, we employ 3D Fourier descriptors and 3D moments for 3D objects and 2D Fourier descriptors for 2D objects. To the best of our knowledge, defining the shape of an object by its complete set of 3D surface points and considering a dynamic object without the assumption of quasi-static equilibrium has not been explored by other authors. Furthermore, to the best of our knowledge, the uses of 3D Fourier descriptors and 3D moments have not been exploited before in the context of deformable manipulation.

Finally, Figure 2.16 illustrates a general overview of our proposed closed-loop visual control system. It incorporates a simulator emulating the dynamics of the soft object (green box), a tracking process to correct the modeled deformation (blue box), and a closed-loop controller (pink box).

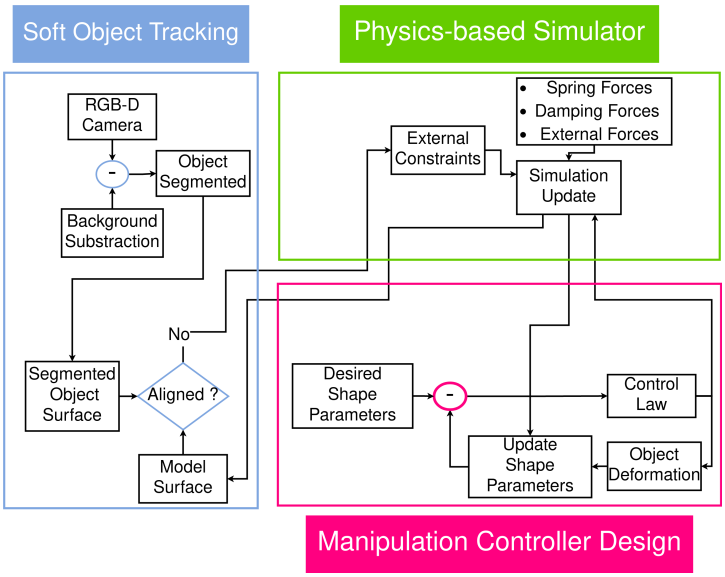


Figure 2.16: Block diagram of the approach: visual tracking part in blue, physics-based simulator in green, and closed-loop control scheme in red.

Chapter 3

Methodology

This chapter presents the derivation of our first approach for deformable object manipulation, which consists of indirectly positioning multiple object points to their desired positions by acting on distant manipulated points. Within this approach, the model points of interest are referred to as feature points, and the object is approximated using a mass-spring model (MSM). An illustration of the desired task is depicted in Figure 3.1. Here, we develop a control law to be applied to the manipulated points to guide the feature points towards their desired positions.

We begin in Section 3.1 by establishing the analytical relationship that relates the displacements of the feature points to the successive motions of the manipulated points. This relationship forms the foundation of our contribution, built upon the MSM while accounting for the propagation delay introduced by it. We start in Section 3.1.1 by formulating the displacements of points within a 1D mesh in response to a series of motions applied to one or two manipulated points. Subsequently, in Section 3.1.2, we expand our formulation scope to encompass a 3D mesh involving multiple manipulated points. Following this, Section 3.1.3 introduces the designed control law.

Then, Section 3.2 presents simulation results that demonstrate the robustness of the proposed closed-loop system. Since our approach is model-based, several factors can influence the efficiency of the control law. One critical consideration is the mesh resolution, which refers to the number of points used to represent the model, which is a significant consideration in any numerical methods. Another important aspect is the model parameters, as it may not always be possible to predict the exact parameters of the soft objects that enable a perfect imitation of the real deformation. To account for these factors, we will include these considerations in our analysis of the proposed approach robustness. Furthermore, this section includes additional results obtained by applying the proposed control law to the system, whether it is fully actuated or not. It also includes comparisons between the proposed approach (PA) and other model-free methods [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018].

Numerical simulations are used to validate the proposed approach for the indirect positioning task, where multiple feature points are considered. In addition, the proposed approach is applicable for the indirect positioning of either single or multiple feature points in real experiments involving real soft objects and robotic manipulators, as outlined in Chapter 4. Furthermore, the pipeline can be adjusted for comprehensive shape servoing, involving the deformation of the entire object surface, as explored in Chapter 5.

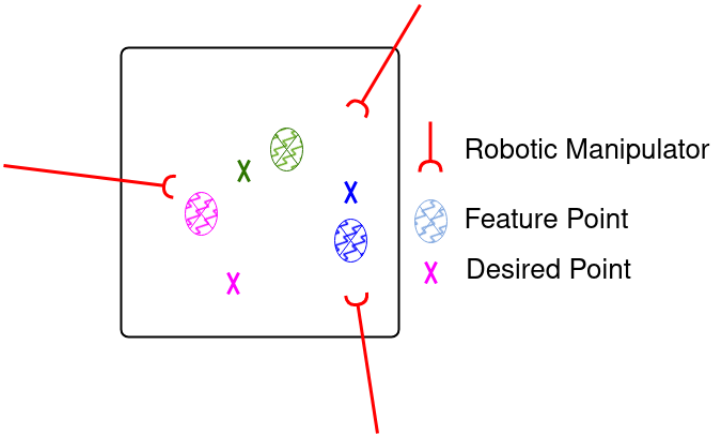


Figure 3.1: An example of indirect positioning of 3 feature points using 3 manipulated points.

3.1 Deformable object servoing

In this modeling section, we assume that the object model is known with its corresponding parameters.

Moreover, the object is modeled using the mass-spring-model (MSM), wherein its dynamic behavior is described by (2.11). In all following analytical derivations related to the modeling and design of the control law, we neglect the external forces $\boldsymbol{f_e}$ since they are a priori unknown. This exclusion includes the gravity force, which, although known, is not considered. This is because the object model is created when the object is in its initial rest state, thereby compensating for both the ground force exerted by the supporting surface and the effects of gravity. However, $\boldsymbol{f_e}$ is involved in the real experiments, presented in Section 4.2.2, and we will see that these external forces do not perturb the stability and robustness of our system. Consequently, (2.11) is simplified to:

$$m_i \ddot{\boldsymbol{x}}_i = \boldsymbol{f_{s_i}} + \boldsymbol{f_{D_i}} \tag{3.1}$$

where $\boldsymbol{f_D}$ represents the damping force with damping value D_v. Regarding $\boldsymbol{f_s}$, it corresponds to the spring force acting on P_i, which depends on the stiffness $\mathbf{K}_{ij}$ connecting P_i to its neighbors, P_j, $\forall j \in \boldsymbol{\nu}_i \subset \mathcal{N}$, and expressed as:

$$\boldsymbol{f_s}_i = \sum_{j \in \boldsymbol{\nu}_i} \mathbf{K}_{ij}(\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - l_{ij}^0)\frac{(\boldsymbol{x}_j - \boldsymbol{x}_i)}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|} = \sum_{j \in \boldsymbol{\nu}_i} \alpha_{ij}\mathbf{K}_{ij}(\boldsymbol{x}_j - \boldsymbol{x}_i) \tag{3.2}$$

Subsequently, we can determine the model point positions at any time $(t + dt)$ by utilizing the semi-implicit Euler integration method, given that we have knowledge of their positions at time (t), with dt representing the simulation time step. We opt for the semi-implicit Euler integration method due to its simplicity and computational efficiency, as detailed in [Bhasin & Liu, 2006]. Finally, the model update is presented as follows:

$$\boldsymbol{x}_{i(t+dt)} = \boldsymbol{x}_{i(t)} + (dt - \frac{dt^2}{m_i}D_v)\dot{\boldsymbol{x}}_{i(t)} + \frac{dt^2}{m_i}\boldsymbol{f_s}_{i(t)} \tag{3.3}$$

with

$$\dot{\boldsymbol{x}}_{i(t)} = (\boldsymbol{x}_{i(t)} - \boldsymbol{x}_{i(t-dt)})/dt \tag{3.4}$$

3.1.1 Simple 1D MSM

An example of a 1D mesh with N nodes is represented in Figure 3.2. In a first part, we consider that the first point, P_1, is manipulated while the other ones obey the dynamics provided in (3.1). In a second part, we will consider the case of two manipulated points.

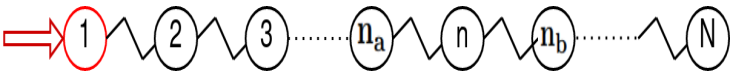


Figure 3.2: An example of a 1D mesh with N nodes connected by springs, including one manipulated point depicted in red.

3.1.1.1 One manipulated point

For illustrative purpose, we first derive the displacements of P_2 due to successive displacements applied on P_1. This will necessitate to also determine the position of P_3. We then generalize the obtained results to determine the position and velocity of any point in the mesh.

Let us consider that P_1 is manipulated such that its successive positions are given by:

$$\begin{aligned} x_{1(t_1)} &= x_{1(t_0)} + \Delta_{1(t_1)} \text{ with } t_1 = t_0 + dt, \\ x_{1(t_2)} &= x_{1(t_1)} + \Delta_{1(t_2)} \text{ with } t_2 = t_1 + dt, \\ x_{1(t_3)} &= x_{1(t_2)} + \Delta_{1(t_3)} \text{ with } t_3 = t_2 + dt, \end{aligned}$$

where $x_{1(t_0)}$ is the position of P_1 in its initial rest state and $\Delta_{1(t)}$ are the successive displacements applied on P_1 at each time (t) (see the red circles in Figure 3.3).

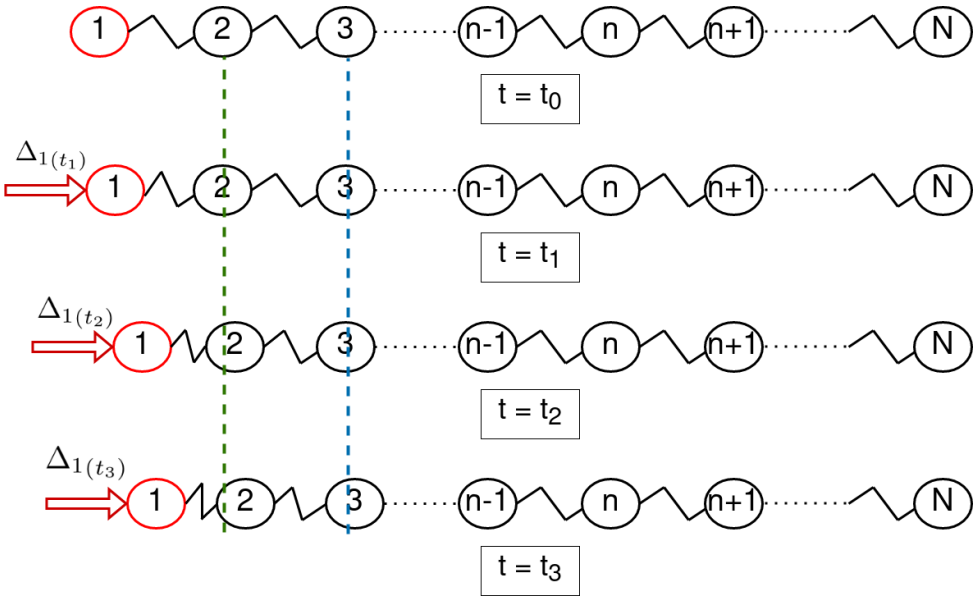


Figure 3.3: An illustration of the propagation delay introduced by the MSM in a 1D mesh when one manipulated point is considered.

From (3.3) and (3.2), the positions of P_2 and P_3 are expressed as follows:

$$x_{2(t+dt)} = x_{2(t)} + (dt - \frac{dt^2}{m_2} D_v)\dot{x}_{2(t)} + \frac{dt^2}{m_2} f_{s21(t)} + \frac{dt^2}{m_2} f_{s23(t)} \tag{3.5}$$

$$x_{3(t+dt)} = x_{3(t)} + (dt - \frac{dt^2}{m_3} D_v)\dot{x}_{3(t)} + \frac{dt^2}{m_3} f_{s32(t)} + \frac{dt^2}{m_3} f_{s34(t)} \tag{3.6}$$

We thus deduce, as illustrated in Figure 3.3, that:

For $t = t_1$: P_2 stays at rest with $x_{2(t_1)} = x_{2(t_0)}$ and $\dot{x}_{2(t_1)} = 0$ since $\dot{x}_{2(t_0)} = f_{s21(t_0)} = f_{s23(t_0)} = 0$. Similarly, P_3 also stays at rest. As shown in Figure 3.3, the MSM induces a propagation delay characterized by the fact that P_2 starts moving at $t_2 = t_1 + dt$ due to the motion applied to P_1 at t_1, whereas P_3 starts moving at time $t_3 = t_1 + 2dt$, etc.

For $t_2 = t_1 + dt$: P_3 is still at rest while P_2 starts moving since $f_{s21(t_1)} \neq 0$. More precisely, we have from (3.2)

$$f_{s21(t_1)} = -K_{12}(l_0 - \Delta_{1(t_1)} - l_0) = K_{12}\Delta_{1(t_1)}$$

from which we obtain:

$$\begin{aligned} x_{2(t_2)} &= x_{2(t_1)} + \gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)}, \\ \text{with } \gamma^{(0)}_{2\,1\,(t_2)} &= \frac{dt^2}{m_2}K_{12} \end{aligned} \tag{3.7}$$

The term $\gamma^{(q)}_{i\,j\,(t)}$ will always appear in the following to represent the propagation coefficient denoting the ratio between the displacement of any node P_i at time t and the displacement of any manipulated point P_j at time $t_1 + qdt$, whatever the mesh and number of manipulated points.

For $t_3 = t_2 + dt$: we obtain from (3.2) and (3.7):

$$\begin{aligned} f_{s21(t_2)} &= -K_{12}[l_0 + \gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)} - (\Delta_{1(t_1)} + \Delta_{1(t_2)}) - l_0] \\ &= K_{12}[(1 - \gamma^{(0)}_{2\,1\,(t_2)})\Delta_{1(t_1)} + \Delta_{1(t_2)}] \qquad (3.8) \\ f_{s23(t_2)} &= K_{23}(2l_0 - (l_0 + \gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)}) - l_0) \\ &= -K_{23}\gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)} \qquad (3.9) \end{aligned}$$

Then, by using (3.5) and (3.7) in (3.4), we get:

$$\begin{aligned} x_{2(t_3)} = x_{2(t_2)} &+ (1 - \frac{dt}{m_2}D_v)\ \gamma^{(0)}_{2\,1\,(t_2)}\ \Delta_{1(t_1)} + \frac{dt^2}{m_2}K_{12}[(1 - \gamma^{(0)}_{2\,1\,(t_2)})\Delta_{1(t_1)} + \Delta_{1(t_2)}] \\ &- \frac{dt^2}{m_2}K_{23}\gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)} \end{aligned} \tag{3.10}$$

which can be rewritten as:

$$x_{2(t_3)} = x_{2(t_1)} + \gamma^{(0)}_{2\,1\,(t_3)}\Delta_{1(t_1)} + \gamma^{(1)}_{2\,1\,(t_2)}\Delta_{1(t_2)} \tag{3.11}$$

with

$$\gamma^{(0)}_{2\,1\,(t_3)} = \gamma^{(0)}_{2\,1\,(t_2)} + (1 - \frac{dt}{m_2} D_v)\gamma^{(0)}_{2\,1\,(t_2)} + \frac{dt^2}{m_2} K_{12}(1 - \gamma^{(0)}_{2\,1\,(t_2)}) - \frac{dt^2}{m_2} K_{23}\gamma^{(0)}_{2\,1\,(t_2)} \tag{3.12}$$

$$\gamma^{(1)}_{2\,1\,(t_2)} = \frac{dt^2}{m_2} K_{12} \tag{3.13}$$

Similarly, since $f_{s32(t_2)} = -f_{s23(t_2)}$, we obtain for P_3:

$$\begin{aligned} x_{3(t_3)} &= x_{3(t_1)} + \frac{dt^2}{m_3} K_{23}\gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)} \\ &= x_{3(t_1)} + \gamma^{(0)}_{3\,1\,(t_3)}\Delta_{1(t_1)} \end{aligned} \tag{3.14}$$

with

$$\gamma^{(0)}_{3\,1\,(t_3)} = \frac{dt^2}{m_3} K_{23}\gamma^{(0)}_{2\,1\,(t_2)}. \tag{3.15}$$

For $t_4 = t_3 + dt$: we obtain by following the same reasoning as before:

$$x_{2(t_4)} = x_{2(t_1)} + \gamma^{(0)}_{2\,1(t_4)}\Delta_{1(t_1)} + \gamma^{(1)}_{2\,1(t_3)}\Delta_{1(t_2)} + \gamma^{(2)}_{2\,1(t_2)}\Delta_{1(t_3)} \tag{3.16}$$

with

$$\begin{aligned} \gamma^{(0)}_{2\,1\,(t_4)} = \gamma^{(0)}_{2\,1\,(t_3)} &+ (1 - \frac{dt}{m_2} D_v)(\gamma^{(0)}_{2\,1\,(t_3)} - \gamma^{(0)}_{2\,1\,(t_2)}) + \frac{dt^2}{m_2} K_{12}(1 - \gamma^{(0)}_{2\,1\,(t_3)}) \\ &- \frac{dt^2}{m_2} K_{23}(\gamma^{(0)}_{2\,1\,(t_3)} - \gamma^{(0)}_{3\,1\,(t_3)}) \end{aligned} \tag{3.17}$$

$$\gamma^{(1)}_{2\,1\,(t_3)} = \gamma^{(1)}_{2\,1\,(t_2)} + (1 - \frac{dt}{m_2} D_v)\gamma^{(1)}_{2\,1\,(t_2)} + \frac{dt^2}{m_2} K_{12}(1 - \gamma^{(1)}_{2\,1\,(t_2)}) - \frac{dt^2}{m_2} K_{23}\gamma^{(1)}_{2\,1\,(t_2)} \tag{3.18}$$

$$\gamma^{(2)}_{2\,1\,(t_2)} = \frac{dt^2}{m_2} K_{12} \tag{3.19}$$

Indeed, we now have using (3.11) and (3.14):

$$\begin{aligned} f_{s21(t_3)} &= -K_{12}[l_0 + \gamma^{(0)}_{2\,1\,(t_3)}\Delta_{1(t_1)} + \gamma^{(1)}_{2\,1\,(t_2)}\Delta_{1(t_2)} - (\Delta_{1(t_1)} + \Delta_{1(t_2)} + \Delta_{1(t_3)}) - l_0] \\ &= K_{12}[\Delta_{1(t_3)} + (1 - \gamma^{(0)}_{2\,1\,(t_3)})\Delta_{1(t_1)} + (1 - \gamma^{(1)}_{2\,1\,(t_2)})\Delta_{1(t_2)}] \\ f_{s23(t_3)} &= K_{23}[2l_0 + \gamma^{(0)}_{3\,1\,(t_3)}\Delta_{1(t_1)} - (l_0 + \gamma^{(0)}_{2\,1\,(t_3)}\Delta_{1(t_1)} + \gamma^{(1)}_{2\,1\,(t_2)}\Delta_{1(t_2)}) - l_0] \\ &= -K_{23}[(\gamma^{(0)}_{2\,1\,(t_3)} - \gamma^{(0)}_{3\,1\,(t_3)})\Delta_{1(t_1)} + \gamma^{(1)}_{2\,1\,(t_2)}\Delta_{1(t_2)}] \end{aligned}$$

which allows getting (3.16) using (3.3) and (3.11) in (3.4) as before.

For any time t written as $t = t_1 + kdt$: we can determine the position of P_2 in function of the successive displacements of P_1 by generalizing the above equations. We obtain:

$$x_{2(t)} = x_{2(t_1)} + \gamma^{(0)}_{2\,1\,(t)}\Delta_{1(t_1)} + \gamma^{(1)}_{2\,1\,(t-dt)}\Delta_{1(t_1+dt)} + ... + \gamma^{(k-1)}_{2\,1\,(t_2)}\Delta_{1(t-dt)} \tag{3.20}$$

where, as general form for $\gamma^{(q)}_{2\,1\,(t)}$:

$$\begin{aligned}\gamma^{(q)}_{2\,1\,(t)} = \gamma^{(q)}_{2\,1\,(t-dt)} + (1 - \frac{dt}{m_2}D_v)\ \delta^{(q)}_{2\,1\,(t-dt)} + \frac{dt^2}{m_2}K_{12}(1 - \gamma^{(q)}_{2\,1\,(t-dt)}) \\ - \frac{dt^2}{m_2}K_{23}(\gamma^{(q)}_{2\,1\,(t-dt)} - \gamma^{(q)}_{3\,1\,(t-dt)})\end{aligned} \tag{3.21}$$

by denoting

$$\delta^{(q)}_{i\,j\,(t)} = \gamma^{(q)}_{i\,j\,(t)} - \gamma^{(q)}_{i\,j\,(t-dt)} \tag{3.22}$$

with $\gamma^{(q)}_{2\,1\,(t_2)} = \frac{dt^2}{m_2}K_{12}, \forall q$ (which is coherent with (3.7), (3.12) and (3.19)) and $\gamma^{(q)}_{2\,1\,(t_1)} = 0$ for initializing the iterative form.

Then, the velocity of P_2 can be deduced by evaluating (3.20) at time t and $t - dt$:

$$\begin{aligned}\dot{x}_{2(t)} &= (x_{2(t)} - x_{2(t-dt)}))/dt \\ &= \delta^{(0)}_{21\,(t)}\dot{x}_{1(t_1)} + \delta^{(1)}_{21\,(t-dt)}\dot{x}_{1(t_2)} + ... + \delta^{(k-1)}_{21\,(t_2)}\dot{x}_{1(t-dt)}\end{aligned}$$

which can be rewritten with the simple form:

$$\dot{x}_{2(t)} = \sum_{q=0}^{k-1} \delta^{(q)}_{2\,1\,(t-qdt)}\ \dot{x}_{1(t_1+qdt)} \tag{3.23}$$

Using the same methodology outlined above, the velocity of any arbitrary point P_n belonging to the 1D mesh presented in Figure 3.2 can be calculated in function of its corresponding propagation coefficient and the motion applied on P_1. We obtain by analogy with (3.23):

$$\dot{x}_{n(t)} = \sum_{q=0}^{k-{}^1r_n} \delta^{(q)}_{n\,1\,(t-qdt)}\dot{x}_{1(t_1+qdt)} \tag{3.24}$$

with $({}^jr_i - 1)$ the number of intermediate points between the manipulated point P_j and the model point P_i (for instance, ${}^1r_2 = 1$ since P_2 is directly attached to P_1 while ${}^1r_3 = 2$),

and, as a general form of the propagation coefficient $\gamma^{(q)}_{n\,1}$ that is involved in $\delta^{(q)}_{n\,1}$:

$$\gamma^{(q)}_{n\,1\,(t)} = \gamma^{(q)}_{n\,1\,(t-dt)} + (1 - \frac{dt}{m_n} D_v)\ \delta^{(q)}_{n\,1\,(t-dt)} + \frac{dt^2}{m_n} K_{n_a n} (\gamma^{(q)}_{n_a\,1\,(t-dt)} - \gamma^{(q)}_{n\,1\,(t-dt)}) - \frac{dt^2}{m_n} K_{n n_b} (\gamma^{(q)}_{n\,1\,(t-dt)} - \gamma^{(q)}_{n_b\,1\,(t-dt)}) \tag{3.25}$$

where P_{n_a} and P_{n_b} are the previous and next neighboring points of P_n (see Figure 3.2) and $\gamma^{(q)}_{n\,1\,(t_n)} = \frac{dt^2}{m_n} K_{n_a n} \gamma^{(q)}_{n_a\,1\,(t_{n-1})}$, with $t_n = t_1 + {}^1r_n dt$. For example, we have $\gamma^{(q)}_{4\,1\,(t_4)} = \frac{dt^2}{m_2}\frac{dt^2}{m_3}\frac{dt^2}{m_4} K_{12} K_{23} K_{34}, \forall q$.

Note that in case it is no more the point P_1 that is manipulated but the other end-point P_N, $\dot{x}_{n(t)}$, $\delta^{(q)}_{n\,N\,(t)}$ and $\gamma^{(q)}_{n\,N\,(t)}$ are obtained by just replacing in (3.24) and (3.25) the terms $\dot{x}_1$, $\delta^{(q)}_{n\,1}$, $\gamma^{(q)}_{n\,1}$, $\gamma^{(q)}_{n_a\,1}$, $\gamma^{(q)}_{n_b\,1}$ and ${}^1r_n$ by $\dot{x}_N$, $\delta^{(q)}_{n\,N}$, $\gamma^{(q)}_{n\,N}$, $\gamma^{(q)}_{n_b\,N}$, $\gamma^{(q)}_{n_a\,N}$ and ${}^N r_n$.

The previous results are valid when only one manipulated point is considered in the 1D mesh. When multiple points are used to deform the object, the velocity of the model points is affected by all the external motions applied. This case is discussed further in the following part.

3.1.1.2 Two manipulated points

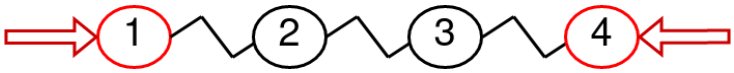


Figure 3.4: An example of a 1D mesh with 4 nodes connected by springs, including two manipulated points depicted in red.

In the example depicted in Figure 3.4, P_1 and P_4 are the manipulated points whose successive positions are given at time $t = t_1 + kdt$ by:

$$\begin{aligned} x_{1(t)} &= x_{1(t_0)} + \sum_{q=0}^{k} \Delta_{1(t_1+qdt)} \\ x_{4(t)} &= x_{4(t_0)} + \sum_{q=0}^{k} \Delta_{4(t_1+qdt)} \end{aligned}$$

At $t_2 = t_1 + dt$, P_2 moves only due to the displacement $\Delta_{1(t_1)}$ of P_1 while P_3 moves only due to the displacement $\Delta_{4(t_1)}$ of P_4 because of the propagation delay introduced by the

MSM. From the results obtained in the previous section, we directly obtain:

$$x_{2(t_2)} = x_{2(t_1)} + \gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)}$$
$$x_{3(t_2)} = x_{3(t_1)} + \gamma^{(0)}_{3\,4\,(t_2)}\Delta_{4(t_1)}$$

where $\gamma^{(0)}_{2\,1\,(t_2)}$ and $\gamma^{(0)}_{3\,4\,(t_2)}$ are obtained from the general form given in (3.25) (using $N = 4$ instead of n and the other permutations mentioned above for computing γ_{34}).

At $t_3 = t_2 + dt$, the position of P_2 changes due to the displacements $\Delta_{1(t_1)}$ and $\Delta_{1(t_2)}$ of P_1 but also due to the displacement $\Delta_{4(t_1)}$ of P_4 that has modified the position of P_3 at t_2 (and vice versa for the position of P_3). This can be seen from the force $f_{s23(t_2)}$ obtained using (3.2) that now contains a supplementary term compared to (3.9):

$$\begin{aligned} f_{s23(t_2)} &= K_{23}[2l_0 + \gamma^{(0)}_{3\,4\,(t_2)}\Delta_{4(t_1)} - (l_0 + \gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)}) - l_0] \\ &= -K_{23}(\gamma^{(0)}_{2\,1\,(t_2)}\Delta_{1(t_1)} - \gamma^{(0)}_{3\,4\,(t_2)}\Delta_{4(t_1)}) \end{aligned}$$

Since the supplementary term is nothing but a sum proportional to $\Delta_{4(t_1)}$, by injecting this force and the other ones in (3.5) and (3.6), we obtain a very similar form as before for the position of P_2 and P_3. More precisely, we obtain:

$$\begin{aligned} x_{2(t_3)} &= x_{2(t_1)} + \gamma^{(0)}_{2\,1\,(t_3)}\Delta_{1(t_1)} + \gamma^{(1)}_{2\,1\,(t_2)}\Delta_{1(t_2)} + \gamma^{(0)}_{2\,4\,(t_3)}\Delta_{4(t_1)} \\ x_{3(t_3)} &= x_{3(t_1)} + \gamma^{(0)}_{3\,4\,(t_3)}\Delta_{4(t_1)} + \gamma^{(1)}_{3\,4\,(t_2)}\Delta_{4(t_2)} + \gamma^{(0)}_{3\,1\,(t_3)}\Delta_{1(t_1)} \end{aligned}$$

By following exactly the same reasoning and computations as in the previous section, it is quite easy to generalize to a 1D mesh whose two endpoints are manipulated. For instance, when P_1 and P_N in Figure 3.2 are deforming the object simultaneously, we obtain the velocity of any point P_n under the form (note the analogy with (3.24)):

$$\dot{x}_{n(t)} = \sum_{q=0}^{k-1 r_n} \delta^{(q)}_{n\,1\,(t-qdt)}\dot{x}_{1(t_1+qdt)} + \sum_{q=0}^{k-N r_n} \delta^{(q)}_{n\,N\,(t-qdt)}\dot{x}_{N(t_1+qdt)} \tag{3.26}$$

What we have developed so far reveals that the model point velocities are connected to the applied manipulated point motions through the proposed propagation coefficients. This holds true whether a single or two manipulated points deform the object simultaneously. In the subsequent paragraph, we explore the general case of a three-dimensional mesh with several manipulated points that deform the object simultaneously.

3.1.2 General 3D MSM

We now consider a general 3D mesh with M manipulators deforming the soft object simultaneously. Let us denote P_{m_l}, $l \in [1, ..., M]$ one of these manipulators. Its motion applied at time $t = kdt$ is given by $\boldsymbol{\Delta}_{m_l(t)} = \dot{\boldsymbol{x}}_{m_l(t)}\, dt$, and we consider $t_1 = 0$ for simplicity.

Following the methodology outlined in the previous section, we first consider a single manipulated point and determine the motion of mesh points in its neighborhood. We then generalize this result to all points in the mesh. Finally, we extend this formulation when all M manipulated points are active. As will be shown below, the main difficulty with respect to the previous 1D case lies in the fact that the forces $\boldsymbol{f_s}_{ij}$ are no more linear with respect to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which will be solved using first-order Taylor decomposition.

3.1.2.1 One manipulated point

For $t = dt$: For any point P_i in the neighborhood of P_{m_l}, we obtain its position using (3.3) as follows:

$$\boldsymbol{x}_{i(dt)} = \boldsymbol{x}_{i(0)} + (dt - \frac{dt^2}{m_i} D_v)\dot{\boldsymbol{x}}_{i(0)} + \frac{dt^2}{m_i} \sum_{j \in \boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij(0)} \tag{3.27}$$

where $\boldsymbol{\nu}_i$ contains two parts: the manipulated point P_{m_l} and the other mesh points P_j in $\boldsymbol{\nu}_i$ ($\boldsymbol{\nu}_i - \{m_l\}$). By exploiting $\boldsymbol{f_s}_{ij}$ in (3.27) we deduce:

$$\sum_{j \in \boldsymbol{\nu}_i - \{m_l\}} \boldsymbol{f_s}_{ij(0)} = \sum_{j \in \boldsymbol{\nu}_i - \{m_l\}} \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)}) \tag{3.28}$$

For all points that do not belong to the neighborhood of P_{m_l}, they remain at their initial positions, and consequently, the spring forces corresponding to these points are zero. This leads to:

$$\sum_{j \in \boldsymbol{\nu}_i - \{m_l\}} \boldsymbol{f_s}_{ij(0)} = \mathbf{0} \tag{3.29}$$

$$\sum_{j \in \boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij(0)} = \boldsymbol{f_s}_{i\,m_l}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{m_l(0)} + \boldsymbol{\Delta}_{m_l(0)}) \tag{3.30}$$

Since $\boldsymbol{f_s}_{i m_l}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{m_l(0)}) = 0$, using a first-order approximation of $\boldsymbol{f_s}_{i\,m_l}$ in (3.30), we obtain:

$$\sum_{j \in \boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij(0)} = \frac{\partial \boldsymbol{f_s}_{i m_l}}{\partial \boldsymbol{x}_{m_l}} \boldsymbol{\Delta}_{m_l(0)} \tag{3.31}$$

where, by denoting $\boldsymbol{d}_{ij} = (\boldsymbol{x}_j - \boldsymbol{x}_i)$, we know from (3.2) that:

$$\frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{x}_j} = \frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{d}_{ij}} \frac{\partial \boldsymbol{d}_{ij}}{\partial \boldsymbol{x}_j} \left(= -\frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{x}_i} \right) = \mathbf{K}_{ij}\mathbf{I}_{3\text{x}3} - \mathbf{K}_{ij} l_0 \frac{\|\boldsymbol{d}_{ij}\|^2 \mathbf{I}_{3\text{x}3} - \boldsymbol{d}_{ij}\boldsymbol{d}_{ij}^T}{\|\boldsymbol{d}_{ij}\|^3} \tag{3.32}$$

Therefore, using (3.31) in (3.27), we obtain:

$$\boldsymbol{x}_{i(dt)} = \boldsymbol{x}_{i(0)} + \frac{dt^2}{m_i} \frac{\partial \boldsymbol{f_s}_{i\,m_l}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{m_l(0)})}{\partial \boldsymbol{x}_{m_l}} \boldsymbol{\Delta}_{m_l(0)} = \boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(0)} \tag{3.33}$$

where

$$\boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} = \frac{dt^2}{m_i} \frac{\partial \boldsymbol{f_s}_{i\,m_l}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{m_l(0)})}{\partial \boldsymbol{x}_{m_l}} \tag{3.34}$$

and $\boldsymbol{\gamma}^{(j)}_{i\,m_l(t)}$ denotes the propagation coefficient that relates the motion of P_i relative to $\boldsymbol{\Delta}_{m_l(jdt)}$ of P_{m_l} at any time t.

Furthermore, for the other points $P_j \in \boldsymbol{\nu}_i$ so that ${}^{m_l}r_j = 2$, we have $\boldsymbol{x}_{j_{(dt)}} = \boldsymbol{x}_{j_{(0)}}$ and $\boldsymbol{\gamma}^{(0)}_{j\,m_l(dt)} = \mathbf{0}$.

For $t = 2dt$: we have for P_i in the neighborhood of P_{m_l}:

$$\boldsymbol{x}_{i(2dt)} = \boldsymbol{x}_{i(dt)} + (dt - \frac{dt^2}{m_i} D_v)\dot{\boldsymbol{x}}_{i(dt)} + \frac{dt^2}{m_i} \sum_{j \in \boldsymbol{\nu}_i} \boldsymbol{f_s}_{ij(dt)} \tag{3.35}$$

where, using (3.4) and (3.33):

$$\dot{\boldsymbol{x}}_{i(dt)} = (\boldsymbol{x}_{i(dt)} - \boldsymbol{x}_{i(0)})/dt = \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(0)}/dt \tag{3.36}$$

Concerning $\boldsymbol{\nu}_i$, it is now decomposed in three parts (see Figure 3.5):

Case 1: for $j \in \boldsymbol{\nu}_i - \{m_l\}$ and ${}^{m_l}r_j = 2$, we have:

$$\begin{aligned} \boldsymbol{f_s}_{ij(dt)} &= \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(0)}, \boldsymbol{x}_{j(0)}) \\ &= \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)}) + \frac{\partial \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)})}{\partial \boldsymbol{x}_i} \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(0)} \\ &= -\frac{\partial \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(0)} \end{aligned} \tag{3.37}$$

Case 2: for $j \in \boldsymbol{\nu}_i - \{m_l\}$ and ${}^{m_l}r_j = 1$, we have:

$$\begin{aligned} \boldsymbol{f}_{\boldsymbol{s}ij\,(dt)} &= \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)}\boldsymbol{\Delta}_{m_l(0)}, \boldsymbol{x}_{j(0)} + \boldsymbol{\gamma}^{(0)}_{j\,m_l(dt)}\boldsymbol{\Delta}_{m_l(0)}) \\ &= \frac{\partial \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)})}{\partial \boldsymbol{x}_j}(\boldsymbol{\gamma}^{(0)}_{j\,m_l(dt)} - \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)})\boldsymbol{\Delta}_{m_l(0)} \end{aligned} \tag{3.38}$$

Case 3: for $j = m_l$, we have:

$$\begin{aligned} \boldsymbol{f}_{\boldsymbol{s}ij\,(dt)} &= \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)}\boldsymbol{\Delta}_{m_l(0)}, \boldsymbol{x}_{j(0)} + \boldsymbol{\Delta}_{m_l(0)} + \boldsymbol{\Delta}_{m_l(dt)}) \\ &= \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)}\boldsymbol{\Delta}_{m_l(0)}, \boldsymbol{x}_{j(0)} + \boldsymbol{\Delta}_{m_l(0)}) + \frac{\partial \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(dt)}, \boldsymbol{x}_{j(dt)})}{\partial \boldsymbol{x}_j}\boldsymbol{\Delta}_{m_l(dt)} \\ &= \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)}) + \frac{\partial \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(dt)}, \boldsymbol{x}_{j(0)} + \boldsymbol{\Delta}_{m_l(0)})}{\partial \boldsymbol{x}_j}\boldsymbol{\Delta}_{m_l(dt)} \\ &\quad + \frac{\partial \boldsymbol{f}_{\boldsymbol{s}ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)})}{\partial \boldsymbol{x}_j}(\boldsymbol{I}_{3\text{x}3} - \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)})\boldsymbol{\Delta}_{m_l(0)} \end{aligned} \tag{3.39}$$

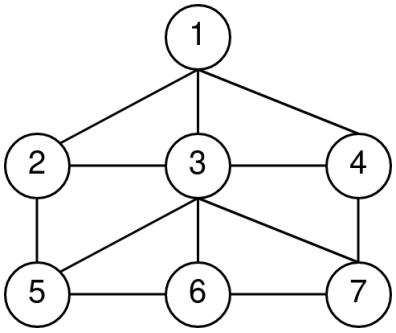


Figure 3.5: An example of a 2D mesh consisting of 7 nodes. By selecting Pm_l as P_1 and P_i as P_3, we have $\boldsymbol{\nu}_i = \{1, 2, 4, 5, 6, 7\}$, ${}^1r_2 = {}^1r_3 = {}^1r_4 = 1$ and ${}^1r_5 = {}^1r_6 = {}^1r_7 = 2$.

By using (3.36), (3.37), (3.38) and (3.39) in (3.35), we obtain:

$$\boldsymbol{x}_{i(2dt)} = \boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(2dt)}\boldsymbol{\Delta}_{m_l(0)} + \boldsymbol{\gamma}^{(1)}_{i\,m_l(dt)}\boldsymbol{\Delta}_{m_l(dt)} \tag{3.40}$$

where:

$$\boldsymbol{\gamma}^{(1)}_{i\,m_l(dt)} = \frac{dt^2}{m_i}\frac{\partial \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(dt)}, \boldsymbol{x}_{m_l(0)} + \boldsymbol{\Delta}_{m_l(0)})}{\partial \boldsymbol{x}_{m_l}} \tag{3.41}$$

$$\begin{aligned}\boldsymbol{\gamma}^{(0)}_{i\,m_l(2dt)} = \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} + (1 - \frac{dt}{m_i}D_v)\boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} + \frac{dt^2}{m_i}\frac{\partial \boldsymbol{f_s}_{im_l}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{m_l(0)})}{\partial \boldsymbol{x}_{m_l}}(\boldsymbol{I}_{3\text{x}3} - \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)}) \\ + \frac{dt^2}{m_i}\sum_{j\in\boldsymbol{\nu}_i - m_l}\frac{\partial \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)})}{\partial \boldsymbol{x}_j}(\boldsymbol{\gamma}^{(0)}_{j\,m_l(dt)} - \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)})\end{aligned} \tag{3.42}$$

since $\boldsymbol{\gamma}^{(0)}_{j\,m_l(dt)} = \boldsymbol{0}$ if ${}^{m_l}r_j = 2$.

Furthermore, for the other points $P_j \in \boldsymbol{\nu}_i$ so that ${}^{m_l}r_j = 2$:

$$\boldsymbol{x}_{j(2dt)} = \boldsymbol{x}_{j(dt)} + (dt - \frac{dt^2}{m_j}D_v)\dot{\boldsymbol{x}}_{j(dt)} + \frac{dt^2}{m_j}\sum_{o\in\boldsymbol{\nu}_j}\boldsymbol{f_s}_{jo(dt)} \tag{3.43}$$

We now have to consider the different cases for $\boldsymbol{f_s}_{jo(dt)}$:

1. if ${}^{m_l}r_o = 3$, we have:
$$\boldsymbol{f_s}_{jo(dt)} = \boldsymbol{f_s}_{jo}(\boldsymbol{x}_{j(0)}, \boldsymbol{x}_{o(0)}) = \boldsymbol{0}$$

2. if ${}^{m_l}r_o = 2$, we have:
$$\boldsymbol{f_s}_{jo(dt)} = \boldsymbol{f_s}_{jo}(\boldsymbol{x}_{j(0)}, \boldsymbol{x}_{o(0)}) = \boldsymbol{0}$$

3. if ${}^{m_l}r_o = 1$, we have:
$$\boldsymbol{f_s}_{jo(dt)} = \boldsymbol{f_s}_{jo}(\boldsymbol{x}_{j(0)}, \boldsymbol{x}_{o(0)} + \boldsymbol{\gamma}^{(0)}_{o\,m_l(dt)}\boldsymbol{\Delta}_{m_l(0)}) = \frac{\partial \boldsymbol{f_s}_{jo}(\boldsymbol{x}_{j(0)}, \boldsymbol{x}_{o(0)})}{\partial \boldsymbol{x}_o}\boldsymbol{\gamma}^{(0)}_{o\,m_l(dt)}\boldsymbol{\Delta}_{m_l(0)}$$

Therefore, by denoting $\boldsymbol{o}$ the set of points belonging to $\boldsymbol{\nu}_j$ so that ${}^{m_l}r_o = 1$, we obtain:

$$\begin{aligned}\boldsymbol{x}_{j(2dt)} &= \boldsymbol{x}_{j(0)} + \frac{dt^2}{m_j}\sum_{o\in\boldsymbol{o}}\frac{\partial \boldsymbol{f_s}_{jo}(\boldsymbol{x}_{j(dt)}, \boldsymbol{x}_{o(0)})}{\partial \boldsymbol{x}_o}\boldsymbol{\gamma}^{(0)}_{o\,m_l(dt)}\boldsymbol{\Delta}_{m_l(0)} \\ &= \boldsymbol{x}_{j(0)} + \boldsymbol{\gamma}^{(0)}_{j\,m_l(2dt)}\boldsymbol{\Delta}_{m_l(0)}\end{aligned} \tag{3.44}$$

Finally, for all other points P_k in the mesh so that ${}^{m_l}r_k > 2$, we have of course $\boldsymbol{x}_{k(2dt)} = \boldsymbol{x}_{k(0)}$.

For any $\boldsymbol{t = kdt}$: From (3.33), (3.35) and (3.44), and by analogy with the 1D case,

we can show that for any model point P_n in the 3D mesh:

$$\boldsymbol{x}_{n(t)} = \hat{\boldsymbol{x}}_{n(t)} + \boldsymbol{\gamma}^{(k-{}^{m_l}r_n)}_{n\,m_l\,({}^{m_l}r_n dt)} \boldsymbol{\Delta}_{m_l(t-{}^{m_l}r_n dt)} \tag{3.45}$$

The intermediate equations that leads to the derivation of (3.45) are provided at the end of this paragraph, where $\hat{\boldsymbol{x}}_{n(t)}$ denotes the position of P_n without considering the last effective displacement $\boldsymbol{\Delta}_{m_l(t-{}^{m_l}r_n dt)}$, which is given by

$$\hat{\boldsymbol{x}}_{n(t)} = \boldsymbol{x}_{n(0)} + \sum_{q=0}^{k-({}^{m_l}r_n+1)} \boldsymbol{\gamma}^{(q)}_{n\,m_l((k-q)dt)} \boldsymbol{\Delta}_{m_l(qdt)} \tag{3.46}$$

and the propagation coefficients are given by

$$\boldsymbol{\gamma}^{(q)}_{n\,m_l(t)} = \boldsymbol{\gamma}^{(q)}_{n\,m_l(t-dt)} + \frac{dt^2}{m_n} \sum_{j\in\boldsymbol{\nu}_n} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k,q)} + (1 - \frac{dt}{m_n} D_v)\, \boldsymbol{\delta}^{(q)}_{n\,m_l(t-dt)} \tag{3.47}$$

with

$$\tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k,q)} = \frac{\partial \boldsymbol{f}_{\boldsymbol{s}nj}(\tilde{\boldsymbol{x}}_{n(({}^{m_l}r_n+q-1)dt)}, \tilde{\boldsymbol{x}}_{j(({}^{m_l}r_n+q-1)dt)})}{\partial \boldsymbol{x}_j} (\boldsymbol{\gamma}^{(q)}_{j\,m_l((k-1-q)dt)} - \boldsymbol{\gamma}^{(q)}_{n\,m_l((k-1-q)dt)}) \tag{3.48}$$

$$\boldsymbol{\delta}^{(q)}_{n\,m_l(t-dt)} = \boldsymbol{\gamma}^{(q)}_{n\,m_l(t-dt)} - \boldsymbol{\gamma}^{(q)}_{n\,m_l(t-2dt)} \tag{3.49}$$

and

1. $\tilde{\boldsymbol{x}}_j = \hat{\boldsymbol{x}}_j$ and $\tilde{\boldsymbol{x}}_n = \boldsymbol{x}_n$ if ${}^{m_l}r_j < {}^{m_l}r_n$
2. $\tilde{\boldsymbol{x}}_j = \hat{\boldsymbol{x}}_j$ and $\tilde{\boldsymbol{x}}_n = \hat{\boldsymbol{x}}_n$ if ${}^{m_l}r_j = {}^{m_l}r_n$
3. $\tilde{\boldsymbol{x}}_j = \boldsymbol{x}_j$ and $\tilde{\boldsymbol{x}}_n = \hat{\boldsymbol{x}}_n$ if ${}^{m_l}r_j > {}^{m_l}r_n$
4. $\boldsymbol{\gamma}^{(q)}_{m_l\,m_l} = \boldsymbol{I}_{3\text{x}3}$, $\forall q$ and $\forall t$

We recall that the index q is related to the displacement applied on P_{m_l} at time qdt. It has to be noted that if $t \leq {}^{m_l}r_n dt$ and $t \leq {}^{m_l}r_j dt$, $\tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k,q)} = \boldsymbol{0}$ because $\boldsymbol{f}_{\boldsymbol{s}nj(t)} = \boldsymbol{0}$. Indeed, we remind that the effect of $\boldsymbol{\Delta}_{m_l(t)}$ applied on P_{m_l} at time t will not take place instantaneously on P_n, but at time ${}^{m_l}t_n$ that depends on the position of P_n with respect to P_{m_l}. For all $t < {}^{m_l}t_n$, we have $\boldsymbol{\gamma}^{(q)}_{n\,m_l(t)} = \boldsymbol{0}$. Finding ${}^{m_l}t_n$ for a 3D mesh is easy since

the mesh can be considered as a graph, with P_{m_l} as a root and its neighbors as the leaves. Thus, ${}^{m_l}t_n$ depends on the number of layers between the root and P_n. If this number is ${}^{m_l}r_n - 1$, then ${}^{m_l}t_n = {}^{m_l}r_n dt$, similarly as for the 1D mesh considered in the previous section.

By comparing (3.47) to (3.25) and exploiting (3.48) and (3.32), we can notice that the form of the propagation coefficient γ_{ij} given in (3.25) for a 1D mesh can be obtained from the form (3.47) obtained for a 3D mesh by neglecting the second term of $\frac{\partial \boldsymbol{f_s}_{ij}}{\partial \boldsymbol{x}_j}$. This implies that the forces $\boldsymbol{f_s}_{ij}$ are linear with respect to $\boldsymbol{x}_j$ and $\boldsymbol{x}_n$, *i.e.*, $\frac{\partial \boldsymbol{f_s}_{nj}}{\partial \boldsymbol{x}_j} = \mathbf{K}_{nj}\mathbf{I}_{3\text{x}3}$.

3.1.2.2 Demonstration of (3.45), (3.46) and (3.47)

In what follows, we demonstrate by induction the recurrence relations (3.45), (3.46) and (3.47).

First, by instantiating (3.46) and (3.45) for $t = dt$ and $t = 2dt$ and for any point P_i within the vicinity of P_{m_l}, which means ${}^{m_l}r_i = 1$, $k = 1$ for $t = dt$, and $k = 2$ for $t = 2dt$, we have:

$$
\begin{aligned}
\hat{\boldsymbol{x}}_{i(dt)} &= \boldsymbol{x}_{i(0)} \\
\boldsymbol{x}_{i(dt)} &= \boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(0)} \qquad (3.50) \\
\hat{\boldsymbol{x}}_{i(2dt)} &= \boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(2dt)} \boldsymbol{\Delta}_{m_l(0)} \\
\boldsymbol{x}_{i(2dt)} &= \hat{\boldsymbol{x}}_{i(2dt)} + \boldsymbol{\gamma}^{(1)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(dt)} \\
&= \boldsymbol{x}_{i(0)} + \boldsymbol{\gamma}^{(0)}_{i\,m_l(2dt)} \boldsymbol{\Delta}_{m_l(0)} + \boldsymbol{\gamma}^{(1)}_{i\,m_l(dt)} \boldsymbol{\Delta}_{m_l(dt)} \qquad (3.51)
\end{aligned}
$$

where (3.50) and (3.51) respectively correspond to (3.33) and (3.40). Then, by instantiating (3.47), we obtain

$$
\boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} = \frac{dt^2}{m_i} \sum_{j \in \boldsymbol{\nu}_i} \frac{\partial \boldsymbol{f_s}_{ij}(\tilde{\boldsymbol{x}}_{i(0)}, \tilde{\boldsymbol{x}}_{j(0)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(0)}_{j\,m_l(0)} \qquad (3.52)
$$

$$
\boldsymbol{\gamma}^{(1)}_{i\,m_l(dt)} = \frac{dt^2}{m_i} \sum_{j \in \boldsymbol{\nu}_i} \frac{\partial \boldsymbol{f_s}_{ij}(\tilde{\boldsymbol{x}}_{i(dt)}, \tilde{\boldsymbol{x}}_{j(dt)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(0)}_{j\,m_l(0)} \qquad (3.53)
$$

$$
\begin{aligned}
\boldsymbol{\gamma}^{(0)}_{i\,m_l(2dt)} &= \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} + (1 - \frac{dt}{m_i} D_v) \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)} \qquad (3.54) \\
&\quad + \frac{dt^2}{m_i} \sum_{j \in \boldsymbol{\nu}_i} \frac{\partial \boldsymbol{f_s}_{ij}(\tilde{\boldsymbol{x}}_{i(0)}, \tilde{\boldsymbol{x}}_{j(0)})}{\partial \boldsymbol{x}_j} (\boldsymbol{\gamma}^{(0)}_{j\,m_l(dt)} - \boldsymbol{\gamma}^{(0)}_{i\,m_l(dt)})
\end{aligned}
$$

At time $t = dt$, the displacement exerted on P_{m_l} impacts only the points P_i in its neighborhood, since all other points remain at rest. Consequently, the only point in $\boldsymbol{\nu}_i$ that has to be considered in $\boldsymbol{f_s}_{ij}$ for (3.52) and (3.53) is $P_j = P_{m_l}$. We thus have $\boldsymbol{\gamma}^{(0)}_{m_l\, m_l(0)} = \boldsymbol{I}_{3\text{x}3}$, $\tilde{\boldsymbol{x}}_{i(0)} = \boldsymbol{x}_{i(0)}$, $\tilde{\boldsymbol{x}}_{m_l(0)} = \boldsymbol{x}_{m_l(0)}$, $\tilde{\boldsymbol{x}}_{i(dt)} = \boldsymbol{x}_{i(dt)}$ and $\tilde{\boldsymbol{x}}_{m_l(dt)} = \boldsymbol{x}_{m_l(0)} + \boldsymbol{\Delta}_{m_l(dt)}$, from which we deduce:

$$\boldsymbol{\gamma}^{(0)}_{i\, m_l(dt)} = \frac{dt^2}{m_i} \frac{\partial \boldsymbol{f_s}_{i m_l}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{m_l(0)})}{\partial \boldsymbol{x}_{m_l}} \tag{3.55}$$

$$\boldsymbol{\gamma}^{(1)}_{i\, m_l(dt)} = \frac{dt^2}{m_i} \frac{\partial \boldsymbol{f_s}_{i m_l}(\boldsymbol{x}_{i(dt)}, \boldsymbol{x}_{m_l(0)} + \boldsymbol{\Delta}_{m_l(dt)})}{\partial \boldsymbol{x}_{m_l}} \tag{3.56}$$

which respectively correspond to (3.34) and (3.41).

At time $t = 2dt$, by decomposing $\boldsymbol{\nu}_i$ as m_l and $\boldsymbol{\nu}_i - \{m_l\}$, we obtain from (3.54):

$$\begin{aligned}\boldsymbol{\gamma}^{(0)}_{i\, m_l(2dt)} = \boldsymbol{\gamma}^{(0)}_{i\, m_l(dt)} + (1 - \frac{dt}{m_i} D_v)\boldsymbol{\gamma}^{(0)}_{i\, m_l(dt)} + \frac{dt^2}{m_i} \frac{\partial \boldsymbol{f_s}_{i m_l}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{m_l(0)})}{\partial \boldsymbol{x}_{m_l}} (\boldsymbol{I}_{3\text{x}3} - \boldsymbol{\gamma}^{(0)}_{i\, m_l(dt)}) \\ + \frac{dt^2}{m_i} \sum_{j \in \boldsymbol{\nu}_i - m_l} \frac{\partial \boldsymbol{f_s}_{ij}(\boldsymbol{x}_{i(0)}, \boldsymbol{x}_{j(0)})}{\partial \boldsymbol{x}_j} (\boldsymbol{\gamma}^{(0)}_{j\, m_l(dt)} - \boldsymbol{\gamma}^{(0)}_{i\, m_l(dt)})\end{aligned}$$

which corresponds to (3.42).

We have validated (3.45), (3.46) and (3.47) for the points in the vicinity of the manipulated point at $t = dt$ and $t = 2dt$. Now, we consider the points P_{v_i} that are not in the neighborhood of P_{m_l}. The points such that ${}^{m_l}r_{v_i} > 2$ are still at rest for $t = dt$ and $t = 2dt$, so we have just to concentrate on points such that ${}^{m_l}r_{v_i} = 2$. Let us denote $\boldsymbol{v}$ as the set of these points. By instantiating (3.46), (3.45) and (3.47) at $t = dt$ and at $t = 2dt$ for any point P_{v_i} belonging to $\boldsymbol{v}$, we have:

$$\begin{aligned}\hat{\boldsymbol{x}}_{v_i(dt)} &= \boldsymbol{x}_{v_i(dt)} = \boldsymbol{x}_{v_i(0)} \\ \hat{\boldsymbol{x}}_{v_i(2dt)} &= \boldsymbol{x}_{v_i(0)} \\ \boldsymbol{x}_{v_i(2dt)} &= \hat{\boldsymbol{x}}_{v_i(2dt)} + \boldsymbol{\gamma}^{(0)}_{v_i\, m_l(2dt)} \boldsymbol{\Delta}_{m_l(0)} = \boldsymbol{x}_{v_i(0)} + \boldsymbol{\gamma}^{(0)}_{v_i\, m_l(2dt)} \boldsymbol{\Delta}_{m_l(0)}\end{aligned} \tag{3.57}$$

with

$$\boldsymbol{\gamma}^{(0)}_{v_i\, m_l(2dt)} = \frac{dt^2}{m_{v_i}} \sum_{j \in \boldsymbol{\nu}_{v_i}} \tilde{\boldsymbol{f}}_{\boldsymbol{s} v_i j(2,0)}$$

The only points in $\boldsymbol{\nu}_{v_i}$ that need to be considered are the points P_j such that ${}^{m_l}r_j = 1$. We denote the set of these points as $\boldsymbol{o}$. For points that belong to $\boldsymbol{\nu}_v$ but do not belong

to $\boldsymbol{o}$, the force $\boldsymbol{f_s}$ is indeed zero. Therefore, we obtain $\tilde{\boldsymbol{x}}_{o(dt)} = \hat{\boldsymbol{x}}_{o(dt)} = \boldsymbol{x}_{o(0)}$ for $o \in \boldsymbol{o}$ and $\tilde{\boldsymbol{x}}_{v_i(dt)} = \boldsymbol{x}_{v_i(dt)}$ for $v_i \in \boldsymbol{v}$, which implies that:

$$\boldsymbol{\gamma}^{(0)}_{v_i\, m_l(2dt)} = \frac{dt^2}{m_{v_i}} \sum_{o \in \boldsymbol{o}} \frac{\partial \boldsymbol{f_s}_{v_i o}(\boldsymbol{x}_{v_i(dt)}, \boldsymbol{x}_{o(0)})}{\partial \boldsymbol{x}_o} \boldsymbol{\gamma}^{(0)}_{o\, m_l(dt)} \tag{3.58}$$

By injecting (3.58) in (3.57), we directly obtain (3.44), as expected.

So far, we have validated (3.45), (3.46) and (3.47) for all the model points at $t = dt$ and $t = 2dt$.

Let us now assume that the position of any point P_n is given by (3.45) at time $t = kdt$, the goal is to obtain the same form at time $t + dt$. As usual, from (3.3), we have:

$$\boldsymbol{x}_{n(t+dt)} = \boldsymbol{x}_{n(t)} + (1 - \frac{dt}{m_n} D_v)(\boldsymbol{x}_{n(t)} - \boldsymbol{x}_{n(t-dt)}) + \frac{dt^2}{m_n} \sum_{j \in \boldsymbol{\nu}_n} \boldsymbol{f_s}_{nj}(\boldsymbol{x}_{n(t)}, \boldsymbol{x}_{j(t)}) \tag{3.59}$$

Using successive first-order approximations of $\boldsymbol{f_s}_{nj}$, similarly as done in (3.37)-(3.39), we have to distinguish three cases:

Case 1: ${}^{m_l}r_j = {}^{m_l}r_n + 1$ (P_j is further from P_{m_l} than P_i):

$$\begin{aligned}
\boldsymbol{f_s}_{nj}(\boldsymbol{x}_{n(t)}, \boldsymbol{x}_{j(t)}) &= \boldsymbol{f_s}_{nj}(\hat{\boldsymbol{x}}_{n(t)}, \boldsymbol{x}_{j(t)}) + \frac{\partial \boldsymbol{f_s}_{nj}(\hat{\boldsymbol{x}}_{n(t)}, \boldsymbol{x}_{j(t)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(k-{}^{m_l}r_n)}_{n\, m_l\, ({}^{m_l}r_n dt)} \boldsymbol{\Delta}_{m_l(t-{}^{m_l}r_n dt)} \\
&= \boldsymbol{f_s}_{nj}(\hat{\boldsymbol{x}}_{n(t)}, \boldsymbol{x}_{j(t)}) + \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj\,(k+1,k-{}^{m_l}r_n)} \boldsymbol{\Delta}_{m_l((k-{}^{m_l}r_n)dt)} \\
&= \sum_{q=0}^{k-{}^{m_l}r_n} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj\,(k+1,q)} \boldsymbol{\Delta}_{m_l(qdt)}
\end{aligned}$$

which is obtained from (3.48).

Case 2: ${}^{m_l}r_j = {}^{m_l}r_n$ (P_j and P_i are at the same distance of P_{m_l}):

$$\begin{aligned}
\boldsymbol{f_s}_{nj}(\boldsymbol{x}_{n(t)}, \boldsymbol{x}_{j(t)}) &= \boldsymbol{f_s}_{nj}(\hat{\boldsymbol{x}}_{n(t)}, \hat{\boldsymbol{x}}_{j(t)}) \\
&\quad + \frac{\partial \boldsymbol{f_s}_{nj}(\hat{\boldsymbol{x}}_{n(t)}, \hat{\boldsymbol{x}}_{j(t)})}{\partial \boldsymbol{x}_j} (\boldsymbol{\gamma}^{(k-{}^{m_l}r_n)}_{j\, m_l\, ({}^{m_l}r_n dt)} - \boldsymbol{\gamma}^{(k-{}^{m_l}r_n)}_{n\, m_l\, ({}^{m_l}r_n dt)}) \boldsymbol{\Delta}_{m_l(t-{}^{m_l}r_n dt)} \\
&= \sum_{q=0}^{k-{}^{m_l}r_n} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj\,(k+1,q)} \boldsymbol{\Delta}_{m_l(qdt)}
\end{aligned}$$

Case 3: ${}^{m_l}r_j = {}^{m_l}r_n - 1$ (P_j is closer from P_{m_l} than P_i):

$$\begin{aligned}\boldsymbol{f}_{\boldsymbol{s}nj}(\boldsymbol{x}_{n(t)}, \boldsymbol{x}_{j(t)}) &= \boldsymbol{f}_{\boldsymbol{s}nj}(\boldsymbol{x}_{n(t)}, \hat{\boldsymbol{x}}_{j(t)}) + \frac{\partial \boldsymbol{f}_{\boldsymbol{s}nj}(\boldsymbol{x}_{n(t)}, \hat{\boldsymbol{x}}_{j(t)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(k-{}^{m_l}r_n+1)}_{j\,m_l\quad({}^{m_l}r_j dt)} \boldsymbol{\Delta}_{m_l(t-{}^{m_l}r_n dt+dt)} \\ &= \sum_{q=0}^{k-{}^{m_l}r_n} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k+1,q)} \boldsymbol{\Delta}_{m_l(qdt)} + \frac{\partial \boldsymbol{f}_{\boldsymbol{s}nj}(\boldsymbol{x}_{n(t)}, \hat{\boldsymbol{x}}_{j(t)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(k-{}^{m_l}r_n+1)}_{j\,m_l\quad({}^{m_l}r_j dt)} \boldsymbol{\Delta}_{m_l(t-{}^{m_l}r_n dt+dt)}\end{aligned}$$

By using (3.45), (3.46), (3.47) and the three previous equations in (3.59), we obtain

$$\begin{aligned}\boldsymbol{x}_{n(t+dt)} &= \boldsymbol{x}_{n(0)} + \sum_{q=0}^{k-({}^{m_l}r_n+1)} \boldsymbol{\gamma}^{(q)}_{n\,m_l((k-q)dt)} \boldsymbol{\Delta}_{m_l(qdt)} + \boldsymbol{\gamma}^{(k-{}^{m_l}r_n)}_{n\,m_l\ ({}^{m_l}r_n dt)} \boldsymbol{\Delta}_{m_l((k-{}^{m_l}r_n)dt)} \\ &+ \frac{dt^2}{m_n} \sum_{j\in\boldsymbol{\nu}_n} \delta_{[{}^{m_l}r_j = {}^{m_l}r_n - 1]} \frac{\partial \boldsymbol{f}_{\boldsymbol{s}nj}(\boldsymbol{x}_{n(t)}, \hat{\boldsymbol{x}}_{j(t)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(k+1-{}^{m_l}r_n)}_{j\,m_l\quad({}^{m_l}r_j dt)} \boldsymbol{\Delta}_{m_l((k+1-{}^{m_l}r_n)dt)} \\ &+ (1 - \frac{dt}{m_n} D_v) \sum_{q=0}^{k-{}^{m_l}r_n} \boldsymbol{\delta}^{(q)}_{n\,m_l(t-qdt)} \boldsymbol{\Delta}_{m_l(qdt)} + \frac{dt^2}{m_n} \sum_{j\in\boldsymbol{\nu}_n} \sum_{q=0}^{(k-{}^{m_l}r_n)} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k+1,q)} \boldsymbol{\Delta}_{m_l((q)dt)} \\ &= \boldsymbol{x}_{n(0)} + \left[\boldsymbol{\gamma}^{(0)}_{n\,m_l(kdt)} + (1 - \frac{dt}{m_n} D_v)\, \boldsymbol{\delta}^{(0)}_{n\,m_l(kdt)} + \frac{dt^2}{m_n} \sum_{j\in\boldsymbol{\nu}_i} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k+1,0)} \right] \boldsymbol{\Delta}_{m_l(0)} + \dots \\ &+ \left[\boldsymbol{\gamma}^{(k-{}^{m_l}r_n)}_{n\,m_l\ ({}^{m_l}r_n dt)} + (1 - \frac{dt}{m_n} D_v)\, \boldsymbol{\delta}^{(k-{}^{m_l}r_n)}_{n\,m_l\ ({}^{m_l}r_n dt)} + \frac{dt^2}{m_n} \sum_{j\in\boldsymbol{\nu}_n} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k+1,k-{}^{m_l}r_n)} \right] \boldsymbol{\Delta}_{m_l((k-{}^{m_l}r_n)dt)} \\ &+ \frac{dt^2}{m_n} \sum_{j\in\boldsymbol{\nu}_n} \left[\delta_{[{}^{m_l}r_j = {}^{m_l}r_n - 1]} \frac{\partial \boldsymbol{f}_{\boldsymbol{s}nj}(\boldsymbol{x}_{n(t)}, \hat{\boldsymbol{x}}_{j(t)})}{\partial \boldsymbol{x}_j} \boldsymbol{\gamma}^{(k+1-{}^{m_l}r_n)}_{j\,m_l\quad({}^{m_l}r_j dt)} \right] \boldsymbol{\Delta}_{m_l((k+1-{}^{m_l}r_n)dt)}\end{aligned} \tag{3.60}$$

where $\delta_{[c]} = 1$ if the condition c is true and $\delta_{[c]} = 0$ otherwise.

Recalling that $t = kdt$ and rearranging the terms in (3.60), we obtain as expected:

$$\boldsymbol{x}_{n(t+dt)} = \hat{\boldsymbol{x}}_{n(t+dt)} + \boldsymbol{\gamma}^{(k+1-{}^{m_l}r_n)}_{n\,m_l\quad({}^{m_l}r_n dt)} \boldsymbol{\Delta}_{m_l(t+dt-{}^{m_l}r_n dt)}$$

with, by identifying the terms in (3.60), we get:

$$\hat{\boldsymbol{x}}_{n(t+dt)} = \boldsymbol{x}_{n(0)} + \sum_{q=0}^{k+1-({}^{m_l}r_n+1)} \boldsymbol{\gamma}^{(q)}_{n\,m_l((k-q)dt)} \boldsymbol{\Delta}_{m_l(qdt)}$$

and

$$\boldsymbol{\gamma}^{(q)}_{n\,m_l(t+dt)} = \boldsymbol{\gamma}^{(q)}_{n\,m_l(t)} + \frac{dt^2}{m_n} \sum_{j\in\boldsymbol{\nu}_n} \tilde{\boldsymbol{f}}_{\boldsymbol{s}nj(k+1,q)} + (1 - \frac{dt}{m_n} D_v)\, \boldsymbol{\delta}^{(q)}_{n\,m_l(t)} \tag{3.61}$$

Finally, by identifying $\boldsymbol{\gamma}_{n\,m_l\;\;({}^{m_l}r_n dt)}^{(k+1-{}^{m_l}r_n)}$ as the term associated to $\boldsymbol{\Delta}_{m_l((k+1-{}^{m_l}r_n)dt)}$ in (3.60), we see that

$$\boldsymbol{\gamma}_{n\,m_l\;\;({}^{m_l}r_n dt)}^{(k+1-{}^{m_l}r_n)} = \frac{dt^2}{m_n}\sum_{j\in\boldsymbol{\nu}_n}\left[\delta_{[{}^{m_l}r_j={}^{m_l}r_n-1]}\frac{\partial \boldsymbol{f}_{\boldsymbol{s}nj}(\boldsymbol{x}_{n(t)},\hat{\boldsymbol{x}}_{j(t)})}{\partial \boldsymbol{x}_j}\boldsymbol{\gamma}_{j\,m_l\;\;({}^{m_l}r_j dt)}^{(k+1-{}^{m_l}r_n)}\right], \tag{3.62}$$

which can be written under the form given by (3.61).

Indeed, setting $q = k+1-{}^{m_l}r_n$ and $t = ({}^{m_l}r_n - 1)dt$ in (3.61), and using (3.48), we obtain from (3.61):

$$\boldsymbol{\gamma}_{n\,m_l\;\;({}^{m_l}r_n dt)}^{(k+1-{}^{m_l}r_n)} = \frac{dt^2}{m_n}\sum_{j\in\boldsymbol{\nu}_n}\left[\frac{\partial \boldsymbol{f}_{\boldsymbol{s}nj}(\tilde{\boldsymbol{x}}_{n(kdt)},\tilde{\boldsymbol{x}}_{j(kdt)})}{\partial \boldsymbol{x}_j}(\boldsymbol{\gamma}_{j\,m_l\;\;((^{m_l}r_n-1)dt)}^{(({}^{k+1-{}^{m_l}r_n}))} - \boldsymbol{\gamma}_{n\,m_l\;\;((^{m_l}r_n-1)dt)}^{(({}^{k+1-{}^{m_l}r_n}))})\right] \tag{3.63}$$

Note that $\boldsymbol{\gamma}_{n\,m_l\;\;((^{m_l}r_n-1)dt)}^{(k+1-{}^{m_l}r_n)} = \boldsymbol{\delta}_{n\,m_l\;\;((^{m_l}r_n-1)dt)}^{(k+1-{}^{m_l}r_n)} = \mathbf{0}$ since $\boldsymbol{\gamma}_{n\,m_l\;\;(t)}^{(k+1-{}^{m_l}r_n)} = \mathbf{0}$, $\forall t < {}^{m_l}r_n dt$. Moreover, we can notice that only $\boldsymbol{\gamma}_{n\,m_l\;\;({}^{m_l}r_n-1)dt)}^{(({}^{k+1-{}^{m_l}r_n}))}$ corresponding to P_j with $j\in\boldsymbol{\nu}_n$ such that ${}^{m_l}r_j = ({}^{m_l}r_n - 1)$ are not equal to zero, which corresponds to Case 3 above. As a result, we have $\tilde{\boldsymbol{x}}_{n(kdt)} = \boldsymbol{x}_{n(kdt)}$, $\tilde{\boldsymbol{x}}_{j(kdt)} = \hat{\boldsymbol{x}}_{j(kdt)}$. Since $\boldsymbol{\gamma}_{n\,m_l\;\;((^{m_l}r_n-1)dt)}^{(({}^{k+1-{}^{m_l}r_n}))} = \mathbf{0}$, this allows (3.63) to be presented in the form given in (3.62), which ends our demonstration.

3.1.2.3 Multiple manipulated points

What we have developed so far focused on a single manipulated point. As we showed previously in the 1D scenario and especially in (3.26), if M manipulated points are used to deform the object simultaneously, the motions of the model points can be determined by summing up the effects of the manipulated points using the propagation coefficients. Therefore, we can show by similarity with the 1D case that:

$$\boldsymbol{x}_{n(t+dt)} = \boldsymbol{x}_{n(0)} + \sum_{l=1}^{M}\boldsymbol{\gamma}_{n\,m_l\;\;({}^{m_l}r_n dt)}^{(k-{}^{m_l}r_n)}\boldsymbol{\Delta}_{m_l(t-{}^{m_l}r_n dt)} + \sum_{l=1}^{M}\sum_{q=0}^{k-({}^{m_l}r_n+1)}\boldsymbol{\gamma}_{n\,m_l((k-q)dt)}^{(q)}\boldsymbol{\Delta}_{m_l(qdt)} \tag{3.64}$$

and

$$\dot{\boldsymbol{x}}_{n(t+dt)} = \sum_{l=1}^{M}\sum_{q=0}^{k-{}^{m_l}r_n}\boldsymbol{\delta}_{n\,m_l(t-qdt)}^{(q)}\,\dot{\boldsymbol{x}}_{m_l(qdt)} \tag{3.65}$$

where $\boldsymbol{\gamma}_{n\,m_l}^{(q)}$ and $\boldsymbol{\delta}_{n\,m_l}^{(q)}$ are respectively given by (3.47) and (3.49).

3.1.3 Control scheme

In this section, Q feature points are considered, and the goal is to bring them to some desired positions by acting on M manipulated points through a kinematic controller whose outputs are the velocities of these manipulated points. We denote $[P_{f_1}, P_{f_2}, \ldots, P_{f_Q}]$ the feature points and $[P_{m_1}, P_{m_2}, \ldots, P_{m_M}]$ the manipulated points.

The measurements from a RGB-D camera are provided with a quite low frequency (typically 33 Hz). On the other hand, the model is updated with a higher frequency than the control law to better represent its dynamic behavior. Denoting ρ the ratio between these two frequencies (in practice, we used $dt = 1.5$ ms so that $\rho = 20$), the control law is updated every ρdt, which means that the velocity of the manipulated points remains constant between $h\rho dt$ and $(h+1)\rho dt$, $\forall h \in \mathbb{N}$. Thanks to (3.65), it is possible to determine the link between the velocity of the features points at time $(h+1)\rho dt$ and the velocity of the manipulated points at time $h\rho dt$, from which the control law will be easily designed.

We start again by considering a single manipulated point, P_{m_l}. From (3.65), we have for any feature point P_{f_i}:

$$\dot{\boldsymbol{x}}_{f_i((h+1)\rho dt)} = \sum_{q=0}^{((h+1)\rho - {}^{m_l}r_{f_i} - 1)} \boldsymbol{\delta}^{(q)}_{f_i m_l (t-qdt)} \dot{\boldsymbol{x}}_{m_l(qdt)} \tag{3.66}$$

Furthermore, we know that $\dot{\boldsymbol{x}}_{m_l(t)} = \dot{\boldsymbol{x}}_{m_l(h\rho dt)}$, $\forall t \in [h\rho dt; (h+1)\rho dt[$, then (3.66) can be rewritten as follows:

$$\dot{\boldsymbol{x}}_{f_i((h+1)\rho dt)} = \sum_{q=h\rho}^{((h+1)\rho - {}^{m_l}r_{f_i} - 1)} \boldsymbol{\delta}^{(q)}_{f_i m_l (t-qdt)} \dot{\boldsymbol{x}}_{m_l(h\rho dt)} + \sum_{q=0}^{(h\rho - 1)} \boldsymbol{\delta}^{(q)}_{f_i m_l (t-qdt)} \dot{\boldsymbol{x}}_{m_l(qdt)}$$

which leads to the very simple form:

$$\dot{\boldsymbol{x}}_{f_i((h+1)\rho dt)} = \boldsymbol{A}_{f_i m_l} \dot{\boldsymbol{x}}_{m_l(h\rho dt)} + \boldsymbol{b}_{f_i m_l}$$

with:

$$\boldsymbol{A}_{f_i m_l} = \sum_{q=h\rho}^{((h+1)\rho - {}^{m_l}r_{f_i} - 1)} \boldsymbol{\delta}^{(q)}_{f_i m_l (t-qdt)} \tag{3.67}$$

which combines the effects of the most recent velocity $\dot{\boldsymbol{x}}_{m_l(h\rho dt)}$ applied on the manipulated

point P_{m_l}, and

$$\boldsymbol{b}_{f_i m_l} = \sum_{q=0}^{(h\rho-1)} \boldsymbol{\delta}^{(q)}_{f_i\, m_l (t-qdt)} \dot{\boldsymbol{x}}_{m_l (qdt)} \tag{3.68}$$

which represents the effects of the velocities previously applied on P_{m_l}.

By now considering M manipulated points and Q feature points, we finally obtain:

$$\dot{\boldsymbol{x}}_f = \boldsymbol{A}\, \dot{\boldsymbol{x}}_m + \boldsymbol{b}, \tag{3.69}$$

with $\dot{\boldsymbol{x}}_f = (\dot{\boldsymbol{x}}_{f_1},\ \dot{\boldsymbol{x}}_{f_2}, ..., \dot{\boldsymbol{x}}_{f_Q})$, $\dot{\boldsymbol{x}}_m = (\dot{\boldsymbol{x}}_{m_1},\ \dot{\boldsymbol{x}}_{m_2}, ..., \dot{\boldsymbol{x}}_{m_M})$, and where the coefficients of matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ are given by:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{f_1 m_1} & \boldsymbol{A}_{f_1 m_2} & \dots & \boldsymbol{A}_{f_1 m_M} \\ \boldsymbol{A}_{f_2 m_1} & \boldsymbol{A}_{f_2 m_2} & \dots & \boldsymbol{A}_{f_2 m_M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{f_Q m_1} & \boldsymbol{A}_{f_Q m_2} & \dots & \boldsymbol{A}_{f_Q m_M} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \sum_{l=1}^{M} \boldsymbol{b}_{f_1 m_l} \\ \sum_{l=1}^{M} \boldsymbol{b}_{f_2 m_l} \\ \vdots \\ \sum_{l=1}^{M} \boldsymbol{b}_{f_Q m_l} \end{bmatrix}.$$

Equation (3.69) presents explicitly the velocity of all feature points in function of the motions of the manipulated points deforming the object. $\boldsymbol{A}$ can be considered as the interaction matrix or the deformation Jacobian by similarity with the classical visual servoing framework [Chaumette & Hutchinson, 2008; Lagneau et al., 2020a; Navarro-Alarcon et al., 2013; J. Zhu et al., 2018], while $\boldsymbol{b}$ is a feed-forward term that does not appear in classical shape servoing work. We will see the importance of this term in Sections 3.2.5 and 4.2.2.2, which include simulation and experimental results.

Then, to indirectly position these feature points to their desired positions $\boldsymbol{x}^*_f = (\boldsymbol{x}^*_{f_1},\ \boldsymbol{x}^*_{f_2}, ..., \boldsymbol{x}^*_{f_Q})$ with an exponential decrease so that $\dot{\boldsymbol{x}}_{\boldsymbol{f}} = -\lambda(\boldsymbol{x}_f - \boldsymbol{x}^*_f)$ where $\lambda > 0$, we deduce from (3.69) the closed-loop control law

$$\dot{\boldsymbol{x}}_m = -\lambda \widehat{\boldsymbol{A}}^{+} (\boldsymbol{x}_f - \boldsymbol{x}^*_f) - \widehat{\boldsymbol{A}}^{+} \widehat{\boldsymbol{b}} \tag{3.70}$$

where $\widehat{\boldsymbol{A}}$ is the approximation of $\boldsymbol{A}$, and $\widehat{\boldsymbol{A}}^{+}$ denotes the Moore-Penrose pseudo-inverse of $\widehat{\boldsymbol{A}}$. Regarding $\widehat{\boldsymbol{b}}$, it is the approximation of $\boldsymbol{b}$. In (3.70), we used $\widehat{\boldsymbol{A}}^{+}$ and $\widehat{\boldsymbol{b}}$ instead of $\boldsymbol{A}^{+}$ and $\boldsymbol{b}$ because we consider that the modeling proposed in (3.69) is accurate but requires exact knowledge of the model parameters. However, obtaining the exact parameter values can be challenging, leading us to use model coarse parameters. Since $\boldsymbol{A}$ and $\boldsymbol{b}$ are dependent on these model parameters, we opted for approximations, which explains our choice of $\widehat{\boldsymbol{A}}$

and $\widehat{\boldsymbol{b}}$.

3.1.3.1 Stability analysis

In this paragraph, we explore the stability analysis and convergence of the closed-loop control law proposed in (3.70) through the application of Lyapunov analysis.

In practice, in order to bring the feature points to their desired positions, it is well-known that at least one manipulated point per feature point has to be involved [Wada et al., 2001b], which implies $M \geq Q$. However, the convergence of the system does not solely depend on the aforementioned condition, but also depends on the rank of $\boldsymbol{A}$, which is influenced by the relative initial positions between the manipulated points and the feature points.

Let define a Lyapunov function $\mathcal{L} = 1/2\|\boldsymbol{x}_f - \boldsymbol{x}_f^*\|^2$, and by considering that the desired points are fixed and replacing (3.70) in (3.69), then we obtain:

$$\dot{\mathcal{L}} = -\lambda(\boldsymbol{x}_f - \boldsymbol{x}_f^*)^T \boldsymbol{A}\widehat{\boldsymbol{A}}^+(\boldsymbol{x}_f - \boldsymbol{x}_f^*) + (\boldsymbol{x}_f - \boldsymbol{x}_f^*)^T(\boldsymbol{b} - \boldsymbol{A}\widehat{\boldsymbol{A}}^+\widehat{\boldsymbol{b}}) \tag{3.71}$$

By denoting the second term as $\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = (\boldsymbol{b} - \boldsymbol{A}\widehat{\boldsymbol{A}}^+\widehat{\boldsymbol{b}})$ and assuming it remains bounded during the deformation. Then, a sufficient condition to ensure the global asymptotic stability of the system is given by:

$$\boldsymbol{A}\widehat{\boldsymbol{A}}^+ > 0 \tag{3.72}$$

If both matrices $\boldsymbol{A}$ and $\widehat{\boldsymbol{A}}^+$ are of full rank $3Q$, and $\widehat{\boldsymbol{A}}^+$ is not too coarse, thus condition (3.72) is ensured. Note that when $\boldsymbol{A}\widehat{\boldsymbol{A}}^+ = \boldsymbol{I}_{3Q}$, which needs $\hat{\boldsymbol{A}}$ to be perfectly estimated, the specified pure exponential decoupled decrease is expected if the MSM fits with the true behavior of the object. Otherwise, perturbations in this decrease will occur.

In case the system is underactuated, i.e. $M < Q$, it is possible to demonstrate that the system is locally asymptotically stable if the condition $\hat{\boldsymbol{A}}^+\boldsymbol{A} > 0$ is ensured, which occurs in case $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}^+$ are of full rank $3M$ and, once again, $\hat{\boldsymbol{A}}^+$ is not too coarse. This means that the system will converge if the error between $\boldsymbol{x}_f$ and $\boldsymbol{x}_f^*$ is initially small [Chaumette & Hutchinson, 2008].

Furthermore, when the system is stable, then the positioning error converges to a value $\overline{\boldsymbol{e}}$, with $\overline{\boldsymbol{e}} = \boldsymbol{\epsilon}^*/\lambda$. However, due to the system stability, it will converge to a quasi-static state where $\overline{\boldsymbol{e}} \to \boldsymbol{0}$ because $\boldsymbol{b} \to \boldsymbol{0}$ and $\hat{\boldsymbol{b}} \to \boldsymbol{0}$. This occurs since $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ encapsulate the effects of previous velocities applied to the manipulated points, which are absorbed by the MSM when it reaches a quasi-static state.

Regarding the rank of $\boldsymbol{A}$, it can be analyzed by examining its terms using (3.67). Indeed, $\boldsymbol{A}_{f_i m_l} \neq \boldsymbol{0}$ when $\rho \geq {}^{m_l}r_{f_i} + 1$, while this condition is not ensured when a manipulated point is located far away from a feature point ($\rho < {}^{m_l}r_{f_i} + 1$). Increasing ρ would be a solution, but a compromise has to be found to maintain a real-time performance. Furthermore, if all the manipulated points are located close to a particular feature point P_{f_i} and far away from the other feature points, then the row in $\boldsymbol{A}$ corresponding to P_{f_i} will be of full rank 3 while all other rows in $\boldsymbol{A}$ will only contain zero values, leading to $rank(\boldsymbol{A}) = 3$.

As a conclusion, when $rank(\boldsymbol{A}) < 3Q$ and the initial error $\boldsymbol{x}_f - \boldsymbol{x}_f^*$ is large, the system may reach a local minimum for which the error $\boldsymbol{x}_f - \boldsymbol{x}_f^* \neq \boldsymbol{0}$ at the end of the deformation task. Looking at the rank of $\boldsymbol{A}$ is thus a precious indicator for the feasibility of indirectly positioning feature points.

3.2 Simulation results

The main objective of the proposed controller (3.70) is to minimize the errors between the current and desired positions of the feature points.

In this section, we show some simulation results using two types of objects, a 3D soft parallelepipedal object with either 1542 or 7710 nodes, each of dimension 12.5x12.5x6.25 cm (see Figure 3.6), and a 2D circular object of radius 10 cm with 256 nodes (see Figure 3.10).

The aim of these simulations is to demonstrate the validity and the robustness of the proposed closed-loop system. Additionally, this section includes comparisons between the proposed approach (PA) and two other model-free methods [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018].

To simulate the behavior of the objects, we use $dt = 1.5$ ms with a control scheme rate of 33 Hz, which corresponds to a Realsense RGB-D camera frequency ($\rho = 20$). For all results shown below, the same controller gain $\lambda = 0.9$ has been used. Simulation results are accessible via this online video: [Indirect-positioning-video].

3.2.1 Object parameters

3.2.1.1 3D soft parallelepipedal object

Two models with the same geometry but with a different resolution are used to simulate the same 3D object (see Figure 3.6). Regarding the parameters of these models, they are

selected to ensure elasticity, as discussed in Section 4.1.2, while also guaranteeing stability.

For the low resolution (sparse) model represented using 1542 nodes and depicted in Figure 3.6(a), the mass is considered to be the same for all points: $m_i^s = 0.35g$. Its corresponding stiffness matrix $\boldsymbol{K}_{ij}^s$ depends on three constant values, $\boldsymbol{K}_{ij}^s = diag(k_x^s, k_y^s, k_z^s)$ with $k_x^s = k_y^s = k_z^s = 13N/m$. Regarding the damping, we selected $D_v^s = 2\sqrt{m_i^s k_x^s} = 0.13Ns/m$.

For the dense model (7710 nodes) represented in Figure 3.6(b), the number of points is five times larger. The mass of the object is the same for both models, so the mass of each point of the dense one is $m_i^d = m_i^s/5$. For determining the stiffness $\boldsymbol{K}_{ij}^d = diag(k_x^d, k_y^d, k_z^d)$ of the dense model, external forces are first applied on both models. Indeed, for points that have the same positions in these two models at their rest states, the same displacements of these points are expected when they reach their new equilibrium states. Therefore, the stiffness of the dense model is updated by minimizing the distance between these points using a gradient descent algorithm. The values obtained are $k_x^d = k_y^d = k_z^d = 8N/m$. Finally, for the damping, we selected $D_v^d = 2\sqrt{m_i^d k_x^d} = 0.05Ns/m$.

3.2.1.2 2D circular object

The 2D object is represented using 256 points, with the mass of each point being $m_i = 0.35g$. Moreover, the stiffness of the springs connecting these nodes is chosen to be the same as for the 3D object: $\boldsymbol{K}_{ij} = diag(k_x, k_y)$ with $k_x = k_y = 13N/m$, leading to $D_v = 0.13Ns/m$.

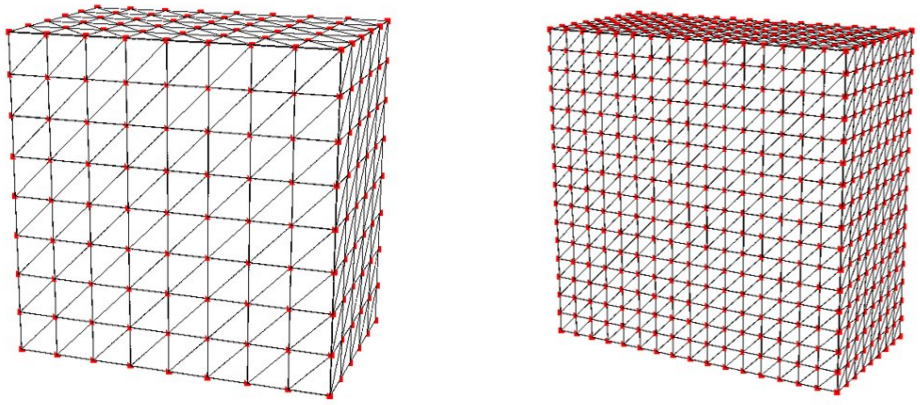
The control law presented in Chapter 3.1.3 is designed for 3D meshes, but it can also be applied to 1D and 2D meshes with minor modifications to account for the different dimensions of the variables in (3.70). For a 3D mesh, $\boldsymbol{A} \in \mathbb{R}^{3Q\text{x}3M}$ and $\boldsymbol{b} \in \mathbb{R}^{3Q}$, while for a 2D mesh, $\boldsymbol{A} \in \mathbb{R}^{2Q\text{x}2M}$ and $\boldsymbol{b} \in \mathbb{R}^{2Q}$.

3.2.2 Robustness analysis – 3D Object

This paragraph discusses four different configurations of the simulated 3D object and how they affect the achievement of the deformation task using the PA.

3.2.2.1 Dense Model vs. Sparse Model

The first configuration consists in studying the influence of the mesh resolution on the deformation results. This is accomplished by examining both dense and sparse models.



(a) (b)

Figure 3.6: Representation of the same 3D object with different mesh resolution: (a) and (b) represent respectively the sparse and the dense mesh.

The system is fully actuated, and 3 feature points are indirectly positioned, *i.e.*, $M = Q = 3$. Moreover, the position of the feature and manipulated points are chosen arbitrary, with the same positions being selected for both models. Two different views of the dense object in its initial rest state and at the end of the deformation task are shown in Figure 3.7. The evolution of the error norm $\|\boldsymbol{x}_f - \boldsymbol{x}_f^*\|$ when using the dense and sparse models is illustrated in Figure 3.8.

As it can be noticed from the orange and blue plots, the error converged in both cases, which demonstrates that the PA is robust to the model resolution. We can just note that the convergence is a bit faster using a higher resolution.

3.2.2.2 Robustness of the controller against model parameters

This paragraph studies the influence of model uncertainties, which occur when dealing with objects whose true physical properties are unknown. For that, the object is simulated using the dense resolution, while the parameters used in the controller differ from those used to simulate the object. More precisely, we consider the cases when the stiffness is half or one and a half times its real value (green and dark blue plots respectively in Figure 3.9), or the damping used in the controller is half or double of its true value (red and pink plots respectively in Figure 3.9), or even a combination of wrong stiffness and damping values (brown plot in Figure 3.9).

From Figure 3.9, the influence of parameter uncertainties can be outlined as follows: when the stiffness is less than its real value, the controller considers that the object is

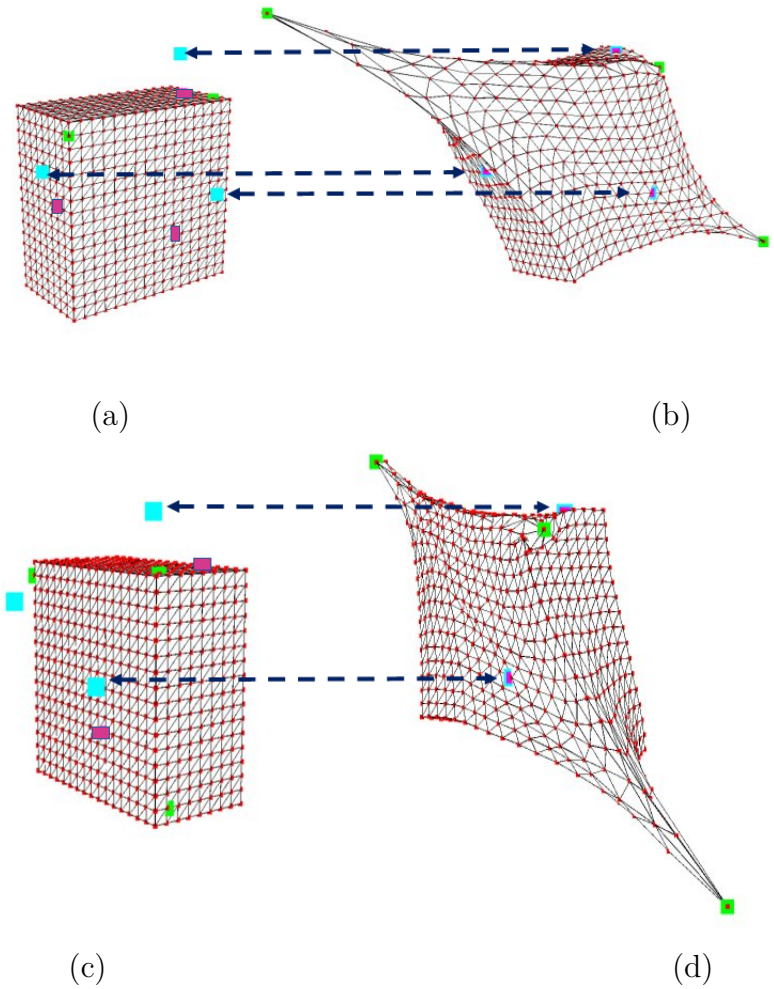


Figure 3.7: Positioning 3 feature points with 3 manipulated points using a dense model: (a) and (c) show the initial configuration for two different views, while (b) and (d) show the final configuration for the same views. The red, green, purple and cyan points are model points, manipulated points, feature points, and their desired positions respectively. The dashed lines link the cyan points corresponding to the same desired points in the initial and final configurations.

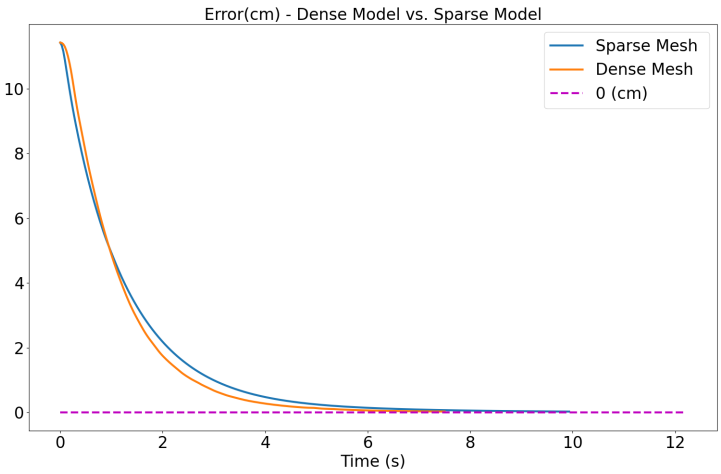


Figure 3.8: Evolution of the positioning error of 3 feature points using 3 manipulated points for two different mesh resolutions.

more elastic and therefore larger control velocities are generated, which makes the error norm converges faster to zero with a small perturbation. This conclusion can be noticed by comparing the orange, green, and dark blue plots. When the damping is less than its real value, the controller considers that less friction is exerted on the object and therefore smaller control velocities are generated, which makes the error norm converges slower to zero (see orange, red, and purple plots). Moreover, by comparing the orange plot with the other plots, we can notice a perturbed exponential decrease in the error norm when coarse parameters were used in the control law, as elaborated in Section 3.1.3.1.

To conclude, the PA is robust to model uncertainties, as evidenced by the convergence of the system across all the aforementioned test cases, as long as the coarse stiffness and damping values used in the controller are not set extremely wrong. It is worth noting that the chosen range of uncertainties is appropriate. For example, if the estimated parameters deviate too significantly from their true values, such as being less than half or more than double the actual values, the simulated model ceases to accurately approximate the real object physical behavior.

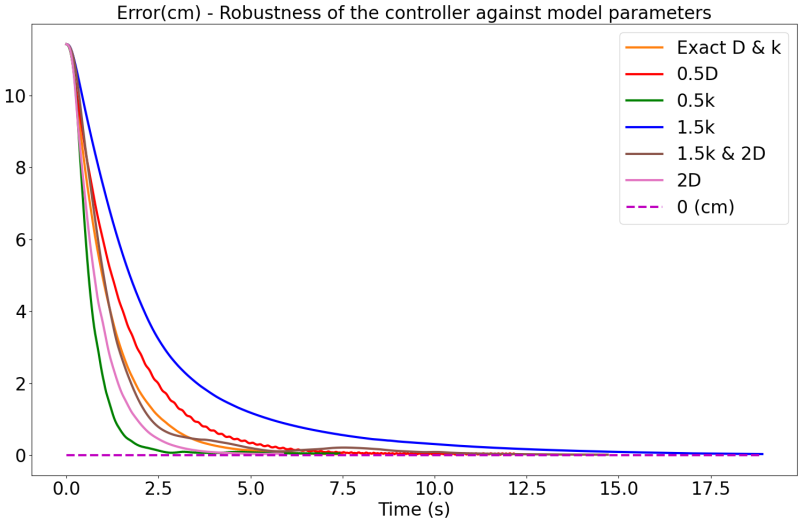


Figure 3.9: Evolution of the positioning error of 3 feature points using 3 manipulated points for testing the PA against wrong model parameters.

3.2.3 Actuation study

This paragraph discusses the impact of the number of manipulated points in relation to the number of feature points.

3.2.3.1 Under-actuated case

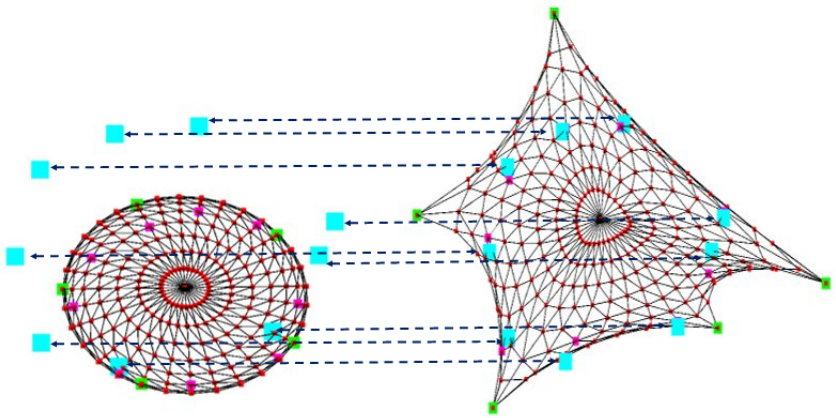
As already said in Section 3.1.3, when the system is under-actuated, *i.e.,* $M < Q$, it may reach a local minimum. In fact, there exist two possibilities.

The first case is a challenging case in which the desired positions of the feature points are chosen arbitrarily. In this case, it may not be possible to reach the specified configuration using the available manipulated points, or due to the physical properties of the object, and a local minimum may be reached.

The second case consists in teaching by showing the desired position of the feature points during an offline teaching step. In this case, a succession of different motions are applied on the chosen manipulated points and the positions of the feature points are saved once the object reaches a new steady state. These saved positions are then considered as the desired positions. Then the model is reset to its initial state and the control scheme is applied to reach again the desired configuration. A similar strategy has already been employed in [Shetab-Bushehri et al., 2022], where a manual deformation was performed to select the desired positions of the feature points. The idea behind this teaching is to

ensure that the desired configuration of feature points is achievable with the employed manipulated points.

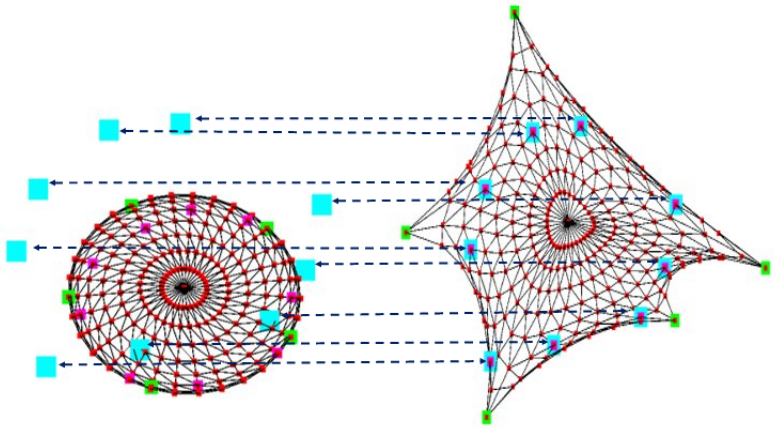
Under-actuated study - 2D object In this experiment, the deformation of a 2D object is attempted using 5 manipulated points distributed around its periphery to indirectly position 9 feature points (see Figure 3.10(a)). The arbitrary selection of the desired 2D positions (cyan points) of the feature points (pink points) leads to the expected convergence to a local minimum, because there are fewer manipulated points than feature points ($M = 9$ while $Q = 5$). This can be shown in Figure 3.10(b) where the actual positions of the feature points do not match their desired positions at the end of the deformation. After updating the desired positions of the feature points to the final positions reached in Figure 3.10(b), the deformation process is repeated once the object is returned to its initial position (see Figure 3.11(a)). In that case, the control scheme is successful as shown in Figure 3.11(b) despite $rank(\boldsymbol{A}) < 3Q$.



(a) (b)

Figure 3.10: Positioning of 9 points of the planar object using 5 manipulated points: the desired positions of the feature points are chosen arbitrary. The red, green, purple and cyan points are model points, manipulated points, feature points, and their desired positions respectively. The dashed lines connect the same desired points.

Under-actuated study - 3D object This experiment aims to deform the 3D object using $M = 2$ and $Q = 3$. The obtained results depicted in Figure 3.12 confirm the outcomes from the earlier 2D case study. The cyan plot indicates that selecting the desired positions of the feature points arbitrarily results in the error norm converging to



(a) (b)

Figure 3.11: Positioning of 9 points of the planar object using 5 manipulated points when the desired positions of the feature points are reachable. (a)-(b) views, colored points and dashed lines defined as in caption of Figure 3.10.

a nonzero value. However, for the teaching case (learned desired position) represented by the black plot, the error converges to zero despite the under-actuated situation.

3.2.3.2 Over-actuated case

In this section, we discuss the cases when there are more manipulated points than feature points, *i.e* $M > Q$.

Over-actuated study – 2D object In this part, we consider more manipulated points than feature points. It allows us to highlight the correct management of the coupling in the system, illustrated by $\boldsymbol{\gamma}_{n\,m_l}$ in (3.65), since each manipulated point influences the position of several feature points.

For the first experiment (see Figure 3.13), 11 points manipulate the object simultaneously, and they are uniformly distributed along its periphery. Meanwhile, only 4 feature points are selected on one side of the object. This selection not only shows that some manipulated points act on the same feature point, but also highlights the influence of the manipulated points that are close to the feature points versus those that are farther away. The success of the deformation task is depicted in Figure 3.13(b), where the feature points (pink points) have converged to their desired positions (cyan points).

For the second experiment (see Figure 3.14), the main objective is to test the amount

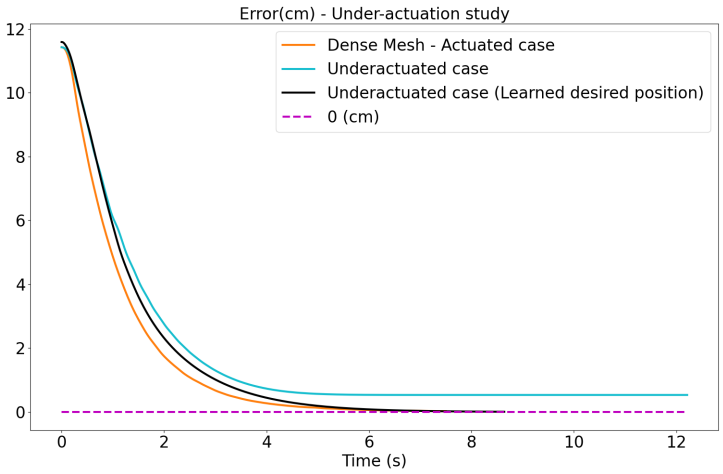
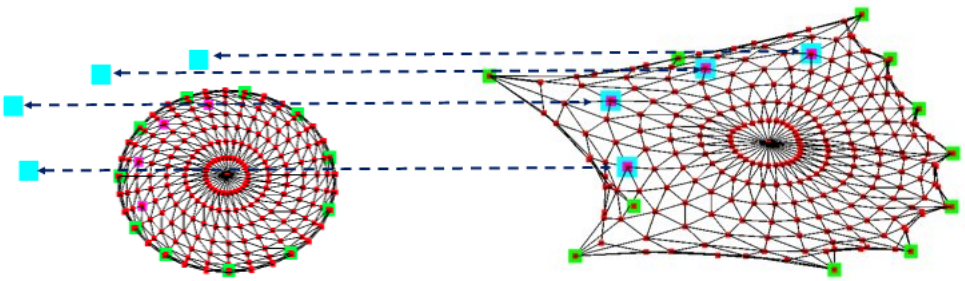


Figure 3.12: Evolution of the positioning error of 3 feature points and two sets of manipulated points, one with 2 and the other with 3 points.

of deformation in function of the number of manipulated points. In this experiment, we either manipulate 5 or 9 points to position 5 feature points. As expected, the object experiences more stretching with a lower number of manipulated points, as can be seen by comparing Figures 3.14(b) and (c).



(a) (b)

Figure 3.13: Positioning of 4 points located on one side of the planar object using 11 manipulated points. (a)-(b) views, colored points and dashed lines defined as in caption of Figure 3.10.

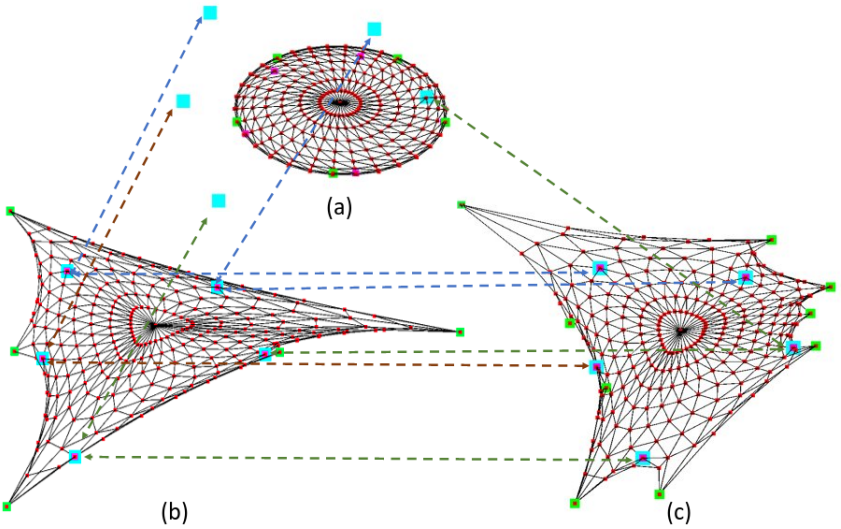


Figure 3.14: Positioning of 5 feature points of the planar object : (a) shows the initial configuration, while (b) and (c) show the final configuration of the deformation task when using 5 and 9 manipulated points, respectively. The colored points and dashed lines are defined as in the caption of Figure 3.10.

3.2.3.3 Over-actuated study - 3D object

In this experiment, we consider exactly the same configuration shown in Figure 3.7 apart 2 supplementary manipulated points have been added. For both cases, as shown on Figure 3.15, the error norm $\|\boldsymbol{x}_f - \boldsymbol{x}_f^*\|$ converges to zero similarly with a nice exponential decrease and with the same time-to-convergence. This result was expected since the same proportional gain λ is used in the control law for both experiments. However, by comparing Figures 3.7 and 3.16, it is clear that when 5 manipulated points are employed, the object exhibits less deformation, which corroborates the results presented in the previous section.

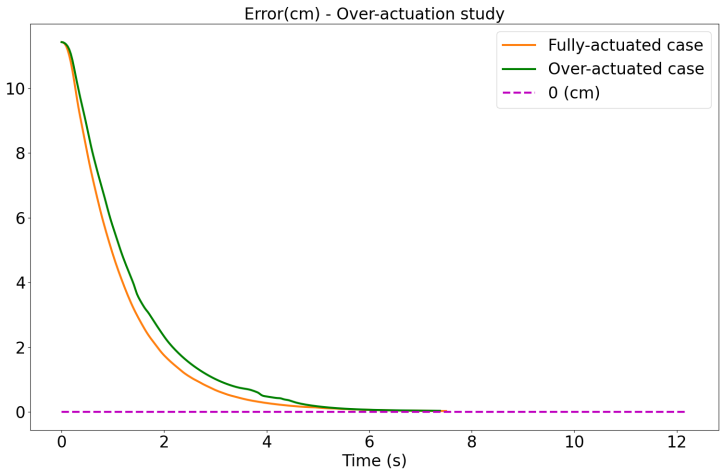


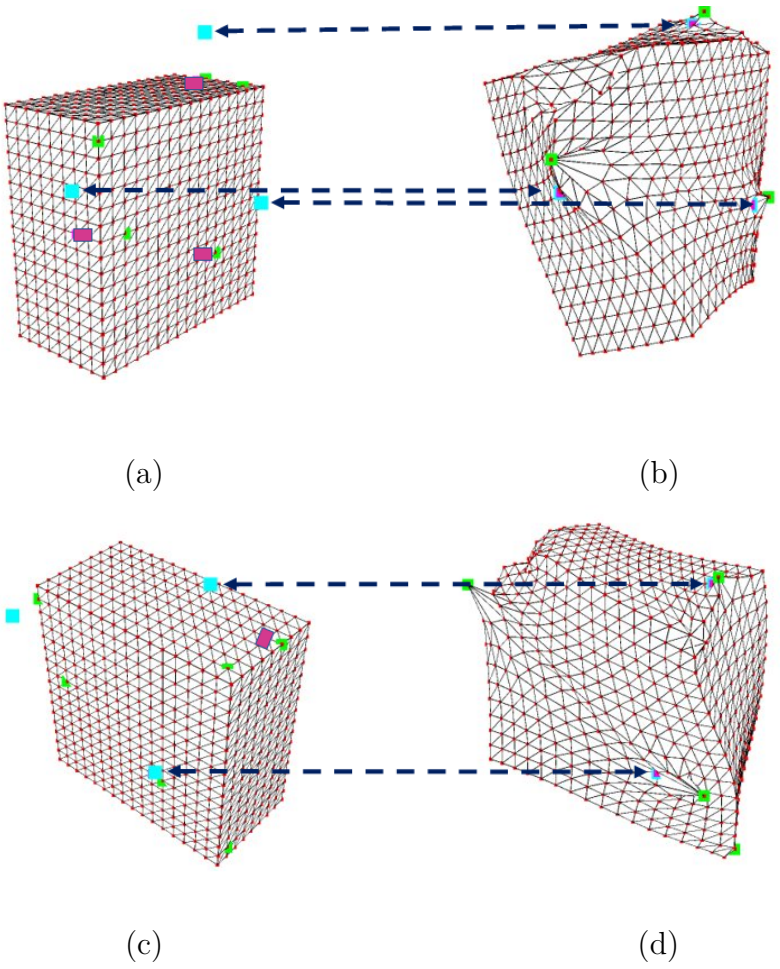
Figure 3.15: Evolution of the positioning error of 3 feature points using PA and two sets of manipulated points, one with 3 and the other with 5 points.

To conclude, the system falls into a local minimum when the number of manipulated points is less than the number of feature points, and arbitrary desired positions are chosen. However, simulation results showed that by teaching the positions of the desired feature points, which respect the geometry and physical properties of the object, the feature points can be driven to their desired positions even when $M < Q$. Moreover, using more manipulated points to position the same feature points results in less deformation and a lower risk of tearing the object.

3.2.4 Comparison with two state-of-the-art approaches

To evaluate the proposed approach, we compare its performance with two model-free approaches [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018] that are both based on an iterative numerical estimation of the deformation Jacobian matrix equivalent to $\boldsymbol{A}$ in (3.69) while neglecting the feed-forward term $\boldsymbol{b}$.

In what follows, the system is considered fully actuated with $M = Q = 3$. Two configurations are considered.



(a) (b)

(c) (d)

Figure 3.16: Positioning 3 feature points with 5 manipulated points: (a)-(d) views, colored points and dashed lines defined as in caption of Figure 3.7.

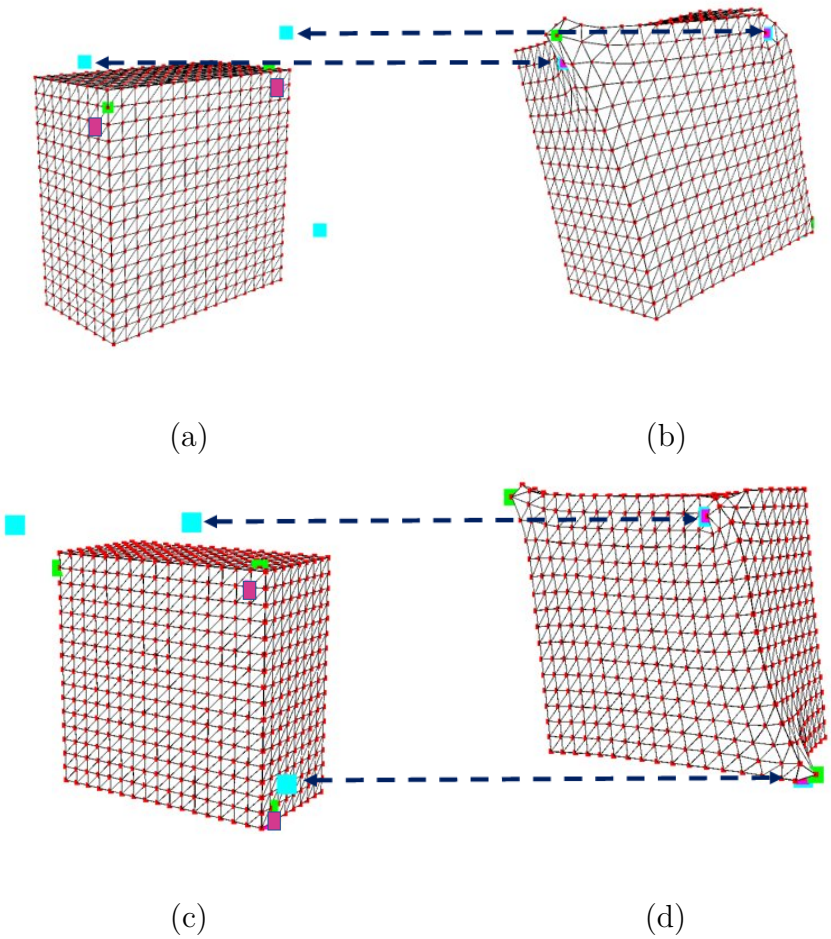
3.2.4.1 First configuration

It consists in using manipulated points in the neighborhood of the feature points, as presented in Figure 3.17. Then, three cases are tested to compare the performance of the positioning task performed with the aforementioned model-free approaches and the PA.

Case 1 – Noise free The first case is an ideal case without measurements noise nor external disturbances. In other meanings, the exact positions of the feature points are fed to the controller. Figure 3.17 shows the configuration of the object in its rest state and after deformation using the PA. As seen in Figure 3.17(b) and (d), the feature points marked in pink are aligned with their desired positions marked in cyan at the end of the deformation. The evolution of the error norm for each approach is illustrated in Figure 3.18. The error norm converges to zero for all the three methods. However, we can note a perfect exponential decrease using the PA that converges faster without any perturbation (red plot).

Case 2 – Noisy data The second case differs from the first one by adding a white noise on the positions of the feature points with a maximum magnitude of 0.4 mm on each axis. It corresponds to the tenth of the distance between two neighboring points of the object mesh when it is in its initial rest state. The aim is to simulate measurement noises of the sensor in charge of providing the current positions of the feature points. Figure 3.19 presents the evolution of the error norm for each method. We can notice that the method proposed in [Navarro-Alarcon et al., 2013] (green plot) fails, while the error norm for our PA (red plot) and the method proposed in [J. Zhu et al., 2018] (blue plot) converge to a steady-state error of about 0.4 mm, which corresponds to the sensor noise norm. Nevertheless, the PA has the advantage of exhibiting an optimal exponential decrease of the error norm.

Case 3 – External perturbations The third case differs from the first by adding for some duration external perturbations to 3 points on the object. This emulates a potential situation in which the object comes into contact with its surroundings, limiting the motion of certain parts of the object that were not anticipated beforehand. The position of the chosen perturbation points is shown in Figure 3.20. Furthermore, these points are periodically displaced with an amplitude of displacement indicated by the cyan plot in Figure 3.21. We can notice in Figure 3.21 that the error norm converges to zero



(a) (b)

(c) (d)

Figure 3.17: First configuration for comparing the PA and the model-free approaches [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018]: (a)-(d) views and colored points defined as in caption of Figure 3.7.

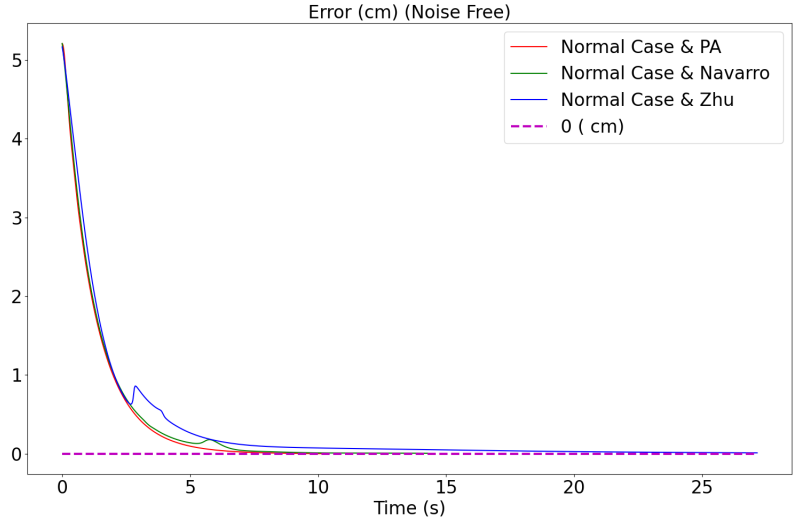


Figure 3.18: Evolution of the error of the positioning task for the case shown in Figure 3.17(a, b) when using the PA and the model-free approaches [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018] in ideal conditions

.

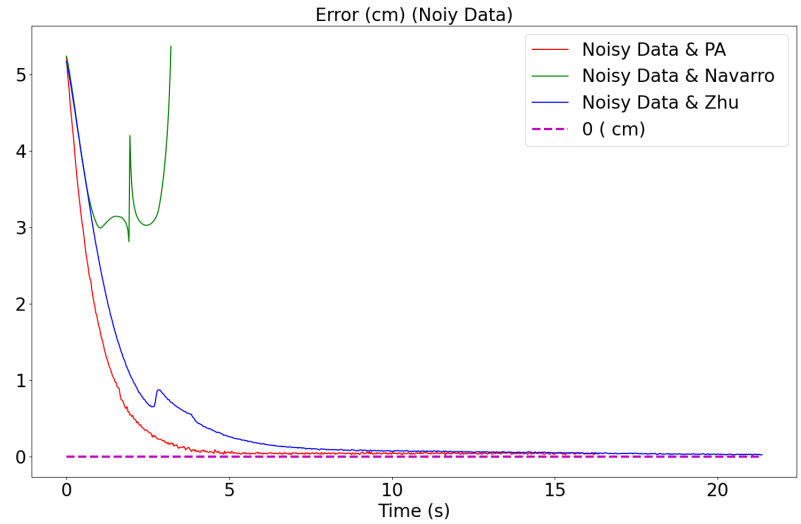
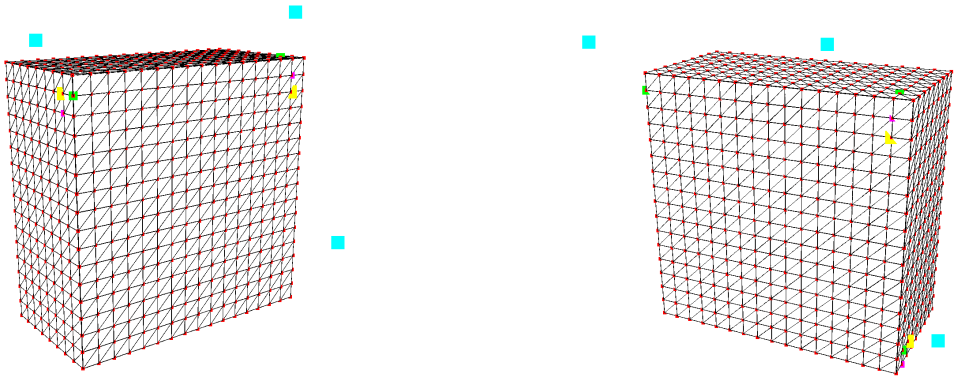


Figure 3.19: Evolution of the error of the positioning task for the case shown in Figure 3.17(a, b) when using the PA and the two model-free approaches [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018] and when noise is added to the measurements of the feature point positions.

at $t = 40\ s$ using the PA and the method proposed in [J. Zhu et al., 2018] (red and blue plots), that is, soon after the perturbation is no more considered ($t > 30\ s$). However, it converges around $t = 80\ s$ for the approach proposed in [Navarro-Alarcon et al., 2013] (green plot). Additionally, during the motion of the perturbation points, that is for $t \leq 30\ s$, the error exhibits less oscillations when using the PA compared to the other approaches. This shows that the PA is less vulnerable to external disturbances.



(a) (b)

Figure 3.20: Same configuration as in Figure 3.17 showing in yellow the position of the perturbation points.

3.2.4.2 Second configuration

It consists in using the same number of feature points as before, while their positions are chosen a bit far from the manipulated points (see Figure 3.22). Figure 3.23 presents the evolution of the error norm for each approach in the case where no measurement noise is added. We can notice the convergence of the error norm using the PA and the method proposed in [Navarro-Alarcon et al., 2013], while the method proposed in [J. Zhu et al., 2018] fails.

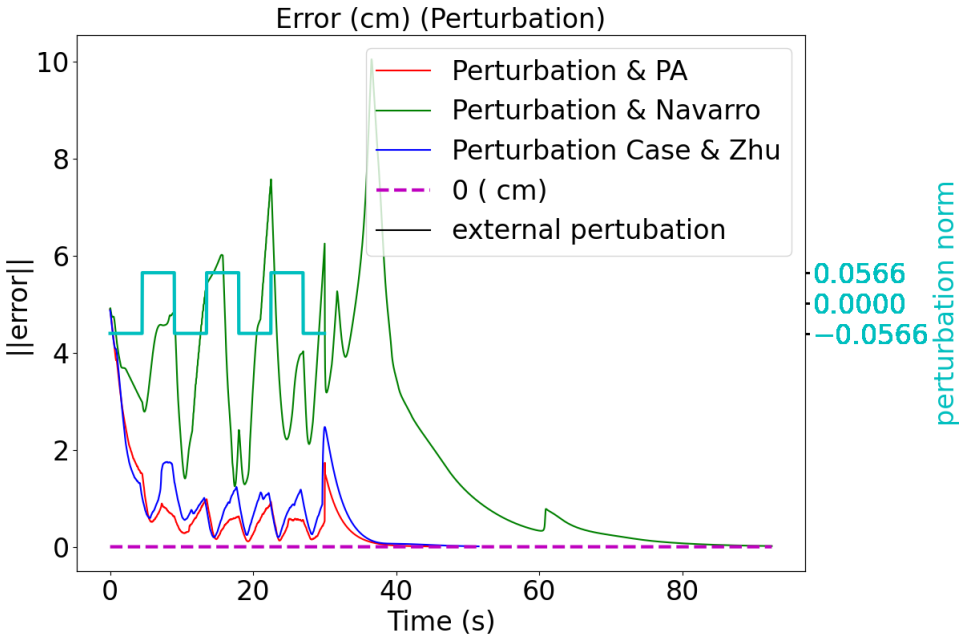
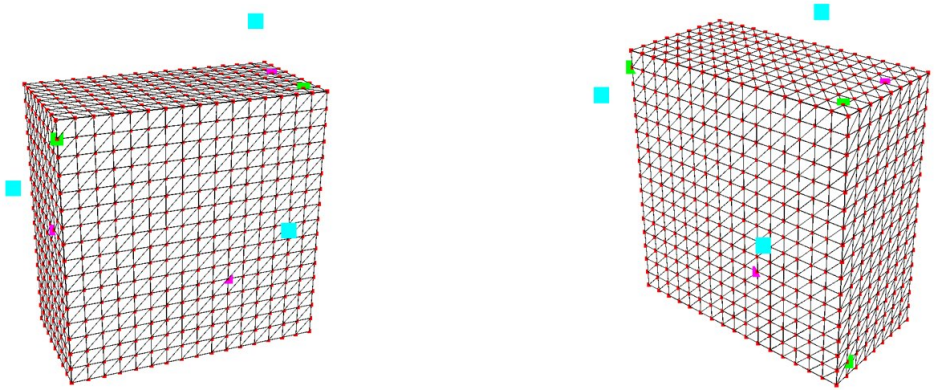


Figure 3.21: Evolution of the error of the positioning task for the case shown in Figure 3.17(a, b) when using the PA and the model-free approaches [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018] and when external perturbations are applied to the yellow points shown in Figure 3.20. The amount of perturbation is illustrated using a cyan plot.



(a) (b)

Figure 3.22: Second configuration for comparing the PA and the model-free approaches [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018], which involves choosing manipulated points a bit far from the feature points: (a)-(b) show two distinct views of the object and colored points defined as in caption of Figure 3.7.

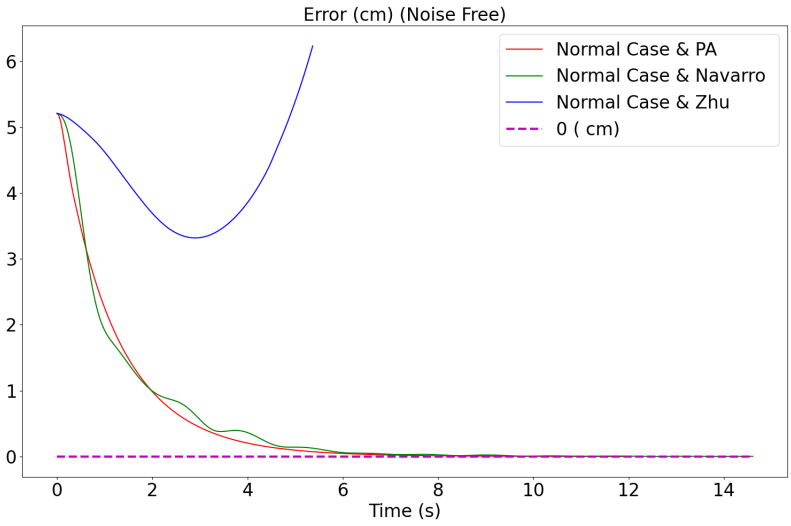


Figure 3.23: Evolution of the error of the positioning task when using the PA and the model-free approaches [Navarro-Alarcon et al., 2013; J. Zhu et al., 2018] in the case where the feature points are far from the manipulated points.

From this comparison study, we found that the PA improves drastically the robustness to measurement noise and external perturbations. Unlike the approach by [Navarro-Alarcon et al., 2013] which struggled in the presence of measurement noise, both our PA and the method proposed by [J. Zhu et al., 2018] demonstrated greater robustness to sensor noise. Furthermore, our PA exhibited reduced sensitivity to external disturbances on soft objects and quickly regained stability after the disturbance ended.
Moreover, both the PA and the method proposed by [Navarro-Alarcon et al., 2013] succeeded in performing the positioning task even when the manipulated points were far from the feature points. However, the method proposed by [J. Zhu et al., 2018] required the manipulated points to be close to the feature points, limiting its effectiveness in such scenarios.

3.2.5 Effect of the feed-forward term

In this paragraph, we analyze the effect of the feed-forward term $\boldsymbol{b}$ involved in the control law (3.70) for the same example of the positioning task shown in Figure 3.17(a,c). We

perform the indirect positioning of the feature points both with and without including this feed-forward term. The evolution of the task error norm is shown in Figure 3.24.

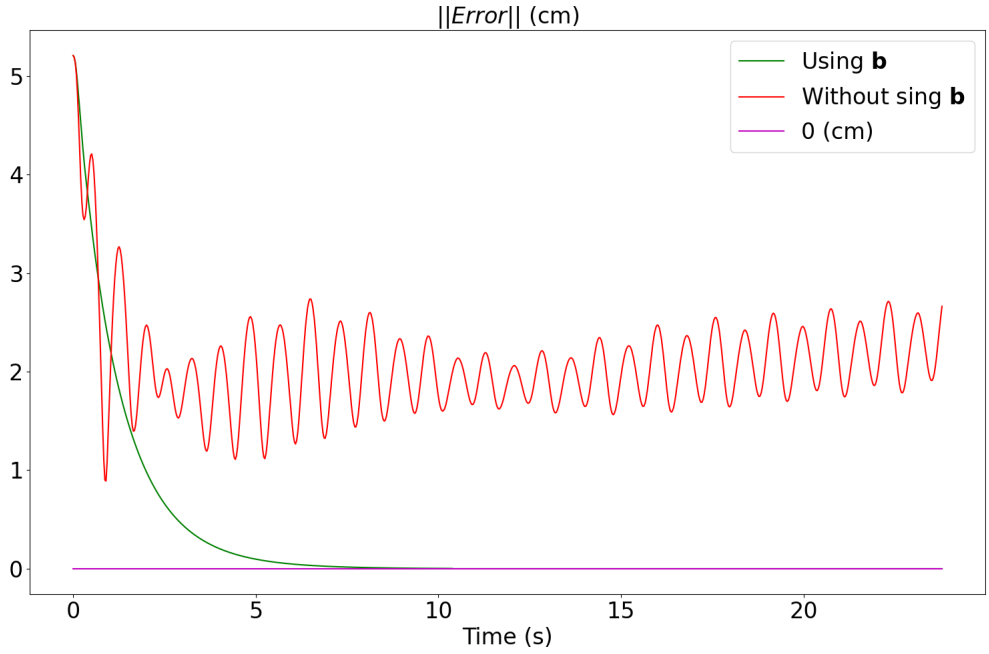


Figure 3.24: Evolution of the error norm of the positioning task shown in Figure 3.17(a,c) with and without the feed-forward term.

As can be seen, without including $\boldsymbol{b}$, the error initially converges but then starts oscillating. This is not the case when $\boldsymbol{b}$ is included, as the error converges to zero without any issues. In conclusion, the feed-forward term plays a crucial role in the system convergence, as its absence leads to system oscillation.

3.3 Conclusion

In this chapter, we presented a comprehensive pipeline detailing the proposed model-based control approach for driving object feature points to desired locations through the control of multiple manipulated points. This pipeline involves approximating spring forces using a first-order Taylor series within a closed-loop system.

We introduced in Section 3.1 a novel analytic formulation of the visual servoing problem utilizing a mass-spring-model. The aim was to establish a relationship that connects the velocities of feature points with the movements of the control points, which is one of the main contributions of this work. We presented intermediary equations to derive

this analytical formulation. We began by illustrating this process using a simplified 1D mesh with a single control point influencing object deformation. This concept was then expanded to encompass multiple control points. Furthermore, we extended the analytical formulation to a general 3D mesh configuration, albeit with only a single control point. Drawing upon insights from the 1D case, we generalized the formulation to encompass multiple control points while adhering to a 3D mesh context.

Then, we provided in Section 3.1.3 simulation results aimed at validating the robustness of the proposed control law with respect to model uncertainties. Additionally, expected behaviors are obtained when the system is fully actuated, under-actuated, and over-actuated. For both the fully actuated and over-actuated cases, the positioning task was achieved without any problem in the tested cases. For the under-actuated case, the positioning error did not necessarily converge to zero when arbitrary desired positions were chosen for the feature points. However, it converged to zero when the desired position were teached during an offline procedure that consists in moving the robot to obtain desired reachable feature point positions.

Thereafter, we compared in Section 3.2.4 the performance of our controller with respect to two model-free approaches and obtained faster convergence, better stability, and less sensitivity to sensor noise and external perturbations. It is important to note that our control law includes an interaction matrix and a feed-forward term, while model-free control laws solely rely on the interaction matrix. This feed-forward term accounts for the dynamics of the model and that is why we did not consider the hypothesis of a quasi-static object, which is the hypothesis held by model-free approaches. To further illustrate the significance of this feed-forward term for system stability, we presented in Section 3.2.5 a simulation result, in which we compared a positioning task with and without this term. Notably, when the feed-forward term was not included, the system exhibited oscillations, whereas its inclusion ensured stable error convergence to zero.

Chapter 4

Indirect Positioning of Multiple Points on a Soft Object

In this chapter, we aim to demonstrate through real experiments the validity of the proposed control law described in the previous chapter. The robotic task is defined as indirectly positioning Q feature points on this object by manipulating M points on the same object.

Since the proposed approach is model-based, we first need to create a model that approximates the physical behavior of the real object. This procedure is detailed in Section 4.1. We begin in Section 4.1.1 by providing a solution to create the object 3D mesh that describes the geometrical part of the object. Then, the object physical behavior is approximated using a simple MSM, and its parameters are estimated as described in Section 4.1.2. However, the MSM provides an approximation of the real object behavior, which in practice may lead to a drift between the real object and its model. To address this issue, an online realignment of the model is performed by tracking the object surface during the deformation using data provided by an RGB-D camera. The object tracking and model correction will be discussed in Section 4.1.3.

Next, in Section 4.2, we present experimental results to demonstrate the effectiveness of the proposed approach (PA). The approach combines vision-based and model-based control. In the previous chapter, we presented different simulation results showing the effectiveness of the PA whether the system is fully actuated or not. In this chapter, we present experimental results and consider both cases, whether the system is fully actuated or over-actuated.

Furthermore, the displacement of object feature points as a function of the applied motions on the manipulated points, as given in (3.69), is divided into two elements: $\boldsymbol{A}$ and $\boldsymbol{b}$. As already explained in the previous chapter, the former term is similar to the interaction matrix in the classical visual servoing framework [Chaumette & Hutchinson, 2008] or the deformation Jacobian in model-free approaches [Lagneau et al., 2020a; Navarro-Alarcon

et al., 2013; J. Zhu et al., 2018]. The latter term is the feed-forward term, which does not exist in the classical visual servoing problems. Therefore, to study its effect in the developed control law (3.70), we will conduct multiple tests both with and without the feed-forward term to further evaluate the effectiveness of this term.

4.1 Deformable object modeling

4.1.1 Model 3D mesh generation

In this paragraph, we present the step-by-step process for constructing the 3D mesh of an unknown object placed on a planar support (typically a table). To achieve this, we utilize one RGB-D camera and the open-source Point Cloud Library (PCL) [Rusu & Cousins, 2011]. If multiple cameras are used, the same processing can be applied, but extrinsic calibration between the cameras is required. The PCL is an open-source library that allows us to work with point clouds and perceive the world in a 3D manner. The RGB-D camera is a 3D sensor that captures color and depth images simultaneously at a high frame rate, typically around 30 Hz, resulting in 3D point clouds. In our work, we use the Realsense D435 camera, which acquires high-resolution large-scale scans consisting of several million 3D points. However, these cameras may introduce various measurement noises, necessitating initial filtering to obtain accurate 3D models. We use several filtering methods proposed by the Realsense library core to improve the data, specifically the decimation filter, fast bilateral filter, temporal filter, and holes filling filter. Once the point cloud is filtered, the data collected by the camera includes not only the deformable object of interest but also its surrounding environment. To focus solely on the deformable object, we need to segment the filtered point cloud.

Since the object is positioned on a flat surface, a hole is created in the plane corresponding to the shadow cast by the object on this surface. We first define the planar surface in the scene using PCL. Then, we compute the convex hull of the plane hole thanks to the method described in [Barber et al., 1996]. In our context, the convex hull represents the smallest convex shape that envelops the boundary of the area where the shadow is cast on the flat surface. Next, we project the points of the point cloud onto the plane, and finally, we determine which projected points lie inside the area enclosed by the convex hull. These points belong to the object sitting on the flat surface.

Given now the set of point cloud corresponding to the segmented object, our objective

is to estimate the object surface by connecting points with each other, ending up with a continuous polygon mesh, such as triangles in our case. The surface triangulation is performed locally by projecting the local neighborhood of a point along the point normal and connecting unconnected points. To compute normals directly from the point cloud, we employ an approximation algorithm called Normalestimation from [Rusu & Cousins, 2011]. This algorithm estimates a point normal from its surrounding points, and we can adjust the level of detail by setting the value of k in the k-neighborhood concept introduced by PCL. However, selecting the appropriate k value is crucial; a too small value might fail to represent a local normal, while a too large value could lead to inefficient computations. In practice, we have used k = 75 for all objects considered.

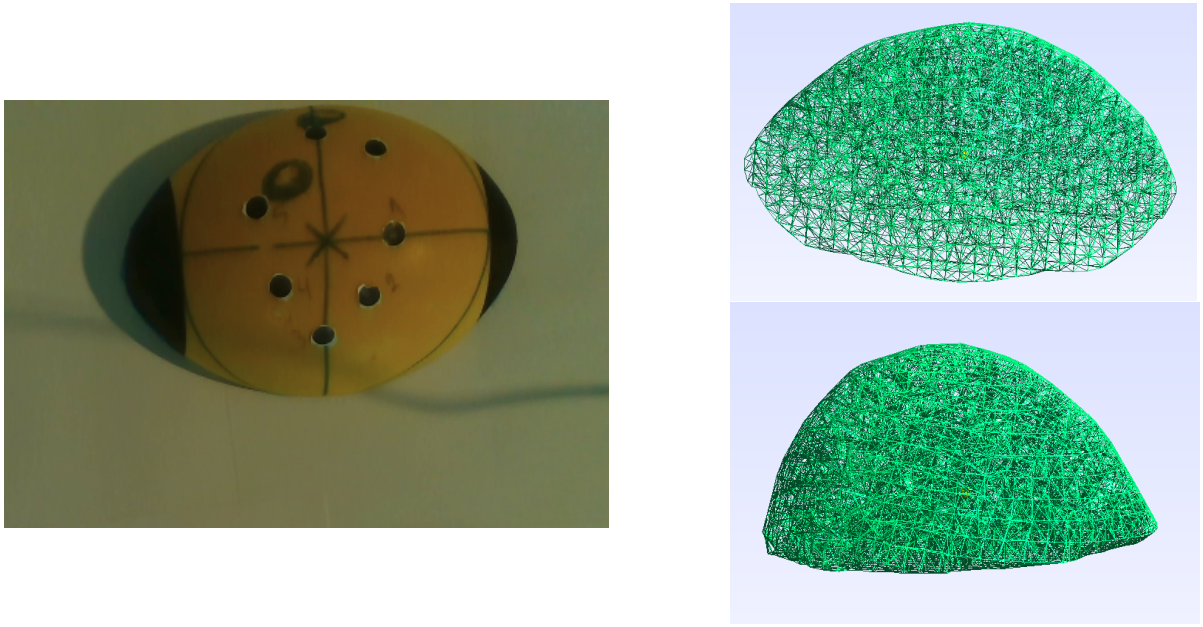
Once the normal estimation is completed, we combine the estimated normals with the object surface coordinates for each point. With these estimated features, we can proceed with the surface reconstruction using the Poisson surface reconstruction method from [Kazhdan et al., 2006]. It is essential to note that our work does not pursue highly accurate reconstruction but rather, to guide a servoing task effectively. The program is written in C++ Builder and utilizes the OpenGL library for 3D display. An example of the constructed mesh is illustrated in Figure 4.1. As can be seen from the reconstructed mesh, the method used is robust against illumination variations.

4.1.2 Model physics parameters estimation

As discussed in Section 2.4, a common approach to estimate the rheological parameters of soft objects involves the following methodology: external forces are applied to the soft object using a robotic manipulator equipped with a force sensor. Then, the same forces are replicated on the object model. Consequently, the model parameters are updated until the deformation observed on the surface of the modeled object aligns with the surface of the real object captured using the camera. We will use the same methodology while employing the MSM.

We recall that the main objective of this work is not to create a perfect model but a simple model sufficient for achieving the shape servoing task. For that purpose, we consider a simple homogeneous MSM.

Regarding the parameters of the MSM used, some of them are imposed by the reconstructed geometrical object mesh: the initial point coordinates, $\boldsymbol{x}_{i(t=0)}$, and the length of springs connecting any two neighboring points, l_{ij}^0, $\forall i \in \mathcal{N}$, $j \in \boldsymbol{\nu}_i$, at rest state. Since the object is supposed to be homogeneous, as stated before, the mass of the object is



(a) (b)

Figure 4.1: An example of mesh reconstruction: (a) corresponds to an ellipsoid ball image obtained from the Realsense D435 camera. (b) represents two views of the constructed mesh of the segmented object.

equally distributed among all points, *i.e.*, $m_i = m_1, \forall i \in \mathcal{N}$.

4.1.2.1 Stiffness matrix estimation

Concerning the spring stiffness $\boldsymbol{K}_{ij}$, it depends on three constant values $\boldsymbol{K}_{ij} = \text{diag}(k_x, k_y, k_z)$, while k_{max} represents the maximum value in $\boldsymbol{K}_{ij}$. This is evident from the work proposed in [Lloyd et al., 2007], which derived an analytical expression for the stiffness of every tetrahedron constituting a mesh and representing it using a MSM. The stiffness is proportional to the volume of the tetrahedron and the Young's modulus of the material properties, which is considered constant according to our hypothesis. Furthermore, during the object deformation, the volume of the tetrahedra constituting the mesh changes, resulting in changes in the stiffnesses of the springs. Based on [Duan et al., 2014], the biggest stiffness corresponding to the largest volume that could appear is chosen to be the same for all springs to prevent collisions between mesh points. At the end, the stiffness matrix is considered constant for the homogeneous object.

We estimate the stiffness matrix by updating it through a gradient descent-based error minimization approach. Our proposed method allows for estimating the stiffness

matrix using either a robotic manipulator equipped with a force sensor or relying solely on vision. In the force-based method, an arbitrary sequence of forces is applied to the real object, and the force sensed by the force sensor is then applied to the object model at the same points where these forces were exerted. Note that these points are not included in the participation of the stiffness matrix update. In the vision-based method, we apply an arbitrary deformation on the real object. Then, some points belonging to the model surface are utilized to induce deformations in the object model. The proportion of these points, in relation to the complete set of surface points, will be analyzed later. Subsequently, these points are moved to be aligned with their corresponding points from the deformation observed on the real object. As for the force-based method, these model points, once aligned, do not participate in updating the stiffness matrix, leaving the remaining points responsible for this update.

Let us denote the set of model surface points as follows: $\boldsymbol{x}_o = [{\boldsymbol{x}_{o_1}}^T, ..., {\boldsymbol{x}_{o_w}}^T]^T,\ \forall\ 1 \leq i \leq w,\ o_i \in \mathcal{N}$ where w is the number of the mesh surface points and o_i is the indices of their corresponding points in $\mathcal{N}$. The segmented surface object using the RGB-D camera is denoted by $\boldsymbol{p}^o$. For every point $\boldsymbol{x}_{o_i} \in \boldsymbol{x}_o, \forall\ 1 \leq i \leq w,\ o_i \in \mathcal{N}$, we denote by $\hat{\boldsymbol{p}}^{o_i} \in \boldsymbol{p}^o$ its correspondent point on the point cloud of the real object. In order to get this correspondence, we apply the same matching technique as in [Petit, Lippiello, Fontanelli, et al., 2017]. Two sets of nearest neighbor correspondences from $\boldsymbol{x}_o$ to $\boldsymbol{p}^o$ and from $\boldsymbol{p}^o$ to $\boldsymbol{x}_o$ are determined. Finally, the correspondence $\hat{\boldsymbol{p}}^{o_i}$ is obtained as a trade-off between these last two correspondences. The discrepancy between the positions of the model points and their actual deformation from the segmented object is employed to update the stiffness matrix. We start by calculating $\mathcal{E}$, the average distance between the two point clouds, $\boldsymbol{x}_o$ and $\boldsymbol{p}^o$:

$$\mathcal{E} = \frac{1}{2w} \sum_{i=1}^{w} \|\boldsymbol{x}_{o_i} - \hat{\boldsymbol{p}}^{o_i}\|^2, \tag{4.1}$$

In practice, the stiffness matrix is well estimated when the error $\mathcal{E}$ is minimized to zero. As a result, the stiffness matrix is adjusted to minimize this error by computing the gradient of the error. We analytically determine this gradient $\nabla\mathcal{E} = \frac{\partial \mathcal{E}}{\partial \boldsymbol{K}_{ij}}$ of the error function (4.1):

$$\boldsymbol{\nabla}\mathcal{E} = \frac{1}{w} \sum_{i=1}^{w} \left[\frac{dt^2}{m_i} \sum_{j \in \boldsymbol{\nu}_{o_i}} \left(\alpha_{o_i j} (\boldsymbol{x}_j - \boldsymbol{x}_{o_i})\right) (\hat{\boldsymbol{p}}^{o_i} - \overline{\boldsymbol{x}}_{o_i}) \right] \tag{4.2}$$

with

$$\overline{\boldsymbol{x}}_{o_i} = \boldsymbol{x}_{o_i} + \frac{dt^2}{m_{o_i}} \sum_{j \in \boldsymbol{\nu}_{o_i}} \alpha_{o_i j} \boldsymbol{K}_{ij} (\boldsymbol{x}_j - \boldsymbol{x}_{o_i}) + (dt - \frac{dt^2}{m_{o_i}} D_v) \dot{\boldsymbol{x}}_{o_i}$$

Finally, the iterative update of the stiffness matrix is obtained as follows:

$$\boldsymbol{K}_{ij} = \boldsymbol{K}_{ij} - 0.5 \boldsymbol{I}_{3\times3} \boldsymbol{\nabla} \mathcal{E} \tag{4.3}$$

with 0.5 in (4.3) denotes the learning rate, and $\boldsymbol{I}_{3\times3}$ the 3×3 identity matrix. We can notice that this calculation does not require a high computation cost, and when $\boldsymbol{x}_o$ and $\boldsymbol{p}^o$ are aligned, the last term in (4.2) will be zero, resulting in a gradient of zero, which ends the update of the stiffness matrix.

Let us note that the simulation operates at a significantly higher frequency than the camera frame rate. As a result, $\boldsymbol{\nabla} \mathcal{E}$ in (4.2) can be calculated multiple times before a new deformation is applied to the object. Consequently, $\boldsymbol{K}_{ij}$ is updated several times for each applied deformation with (4.3). For instance, when employing a typical mesh with around 5k points, $\boldsymbol{K}_{ij}$ undergoes updates between 5 to 10 times for each applied deformation.

Vision-based stiffness estimation In this paragraph, we want to validate the stiffness estimation of a real soft object subjected to deformations from a robotic manipulator, using the approach proposed above. Moreover, the employed robot is not equipped with force sensor, therefore two distinct categories of points emerge in the stiffness matrix estimation. The first category involves some model surface points that are constrained to be aligned with their corresponding positions on the segmented object surface. The second category involves model points that are free to move according to the MSM simulation, which are used to update the stiffness matrix. This involves comparing their positions obtained using the MSM kinematic model (3.3) with its actual stiffness matrix, and their corresponding points on the segmented object surface.

The selection of points within the first category is not arbitrary; it has to be carefully chosen to accurately reflect the deformation characteristics of the object. For example, if deformation primarily occurs on one side of the object, exclusively selecting points belonging to the first category from that side could impede the stiffness matrix update. This is due to the other side representing the static portion of the object, resulting in $\mathcal{E} = \mathbf{0}$ in (4.1). Therefore, it is necessary to strategically select points that encompass a balanced representation of both deforming and non-deforming regions. This meticulous selection process ensures that model refinement aligns faithfully with the real physical be-

havior of the object, consequently fostering precise deformations and accurate mechanical responses.

In our case, the points in both categories are uniformly spread across the object surface. Furthermore, we explore four distinct distributions of these points denoted as y_2, y_3, y_5, and y_{10}. In the context of the first distribution, y_2, for any two points used in the first category, there exists one point that belongs to the second category. Employing a similar approach, we investigate the remaining distributions. The primary objective of these diverse distributions is to assess their impact on the stiffness update process.

In the subsequent results, we present only K_x, the first component of the stiffness matrix, since an identical process is replicated for the remaining elements. We initiate the procedure by using two arbitrary stiffness values: $K_x = 30\ N.m^{-1}$ (see Figure 4.2) and $K_x = 1\ N.m^{-1}$ (see Figure 4.3). The purpose of testing these values is to assess whether the system might become trapped in a local minimum when the initial value is significantly far from the optimal value. Additionally, we aim to determine the number of steps the system takes to converge to a range within the vicinity of the optimal values. It is important to acknowledge that there is not a singular optimal value because the object undergoes a series of deformations. With each deformation, a distinct stiffness value is obtained. For a small deformation, the stiffness required to replicate the physical behavior eventually differs from the stiffness needed to simulate large deformations. Consequently, our objective is to identify a value that proves effective for both minor and substantial deformations.

Subsequently, all deformations are applied to the object, triggering the launch of the stiffness estimation process. The progression of K_x values at each step is illustrated in Figure 4.2 and Figure 4.3. Each step corresponds to a deformation, gradually incrementing until the final step (320), which corresponds to the most significant deformation. Upon completion, indicated by the vertical dashed line in the mentioned figures, the object is returned to its initial state, and the same deformations are applied once more. However, this time, the last estimated stiffness value is adopted as the initial stiffness for the model. Ultimately, the selected stiffness value is determined based on the results obtained in the second configuration (steps > 320).

We notice from Figure 4.2 that in just a few iterations, the stiffness value decreases rapidly, covering a range of values between 1.5 and 4.0 across the different tested distributions. This result is crucial, particularly when starting with an arbitrary initial value, the process quickly reaches a point of agreement. This is important when dealing with

unknown objects because setting an initial guess of the stiffness value would be difficult.

In addition, both Figure 4.2 and Figure 4.3 highlight that the range of estimated values for K_x remains relatively narrow across different deformations (steps) for y_2 and steps > 320. This is because the object behavior is heavily influenced by visual constraints, rather than faithfully representing its physical response. On the other hand, with other distributions, fluctuations in stiffness are observed depending on the applied deformations. Notably, the choice of distributions does not exert a significant influence on stiffness estimation, given that a sufficient number of freely movable points are updated through the MSM. However, it is important to note that this was not the case for the distribution y_2. Leveraging these results, the highest stiffness value attained is selected, yielding $K_x = 3.0\ N.m^{-1}$.

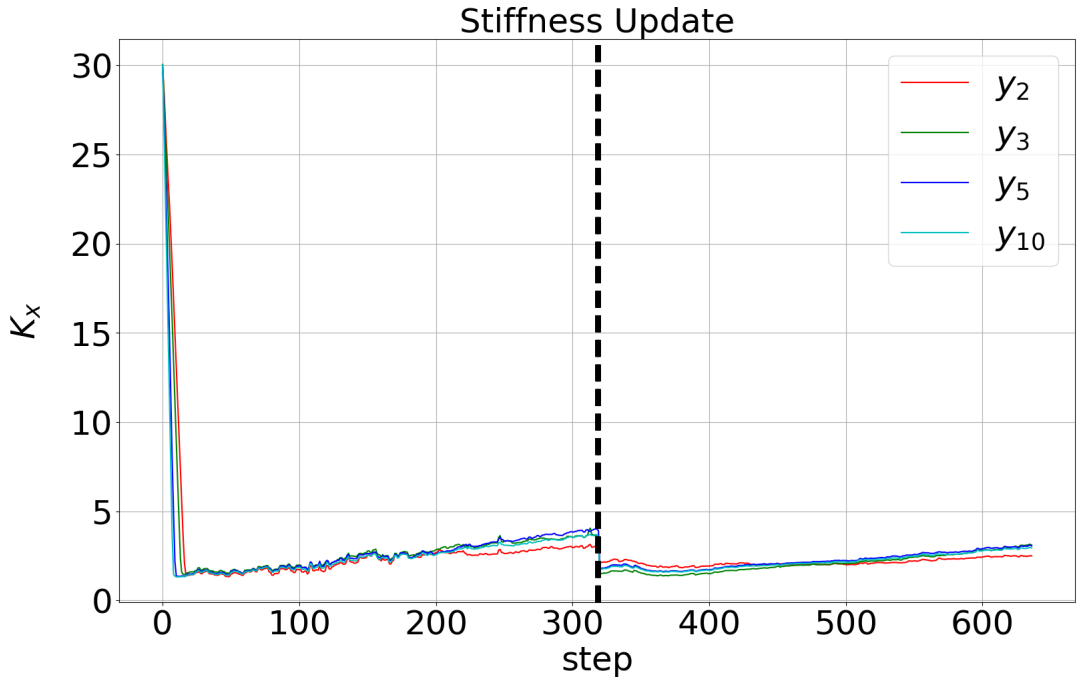


Figure 4.2: Evolution of stiffness updates as a function of applied deformations at each step for each distribution (y_2, y_3, y_5, and y_{10}), starting from an arbitrary initial value of $K_x = 30\ N.m^{-1}$.

4.1.2.2 Damping estimation

Regarding the damping value D_v, it is calculated as presented in [Bhasin & Liu, 2006] in order to guarantee the numerical stability of the system: For that, we have to ensure

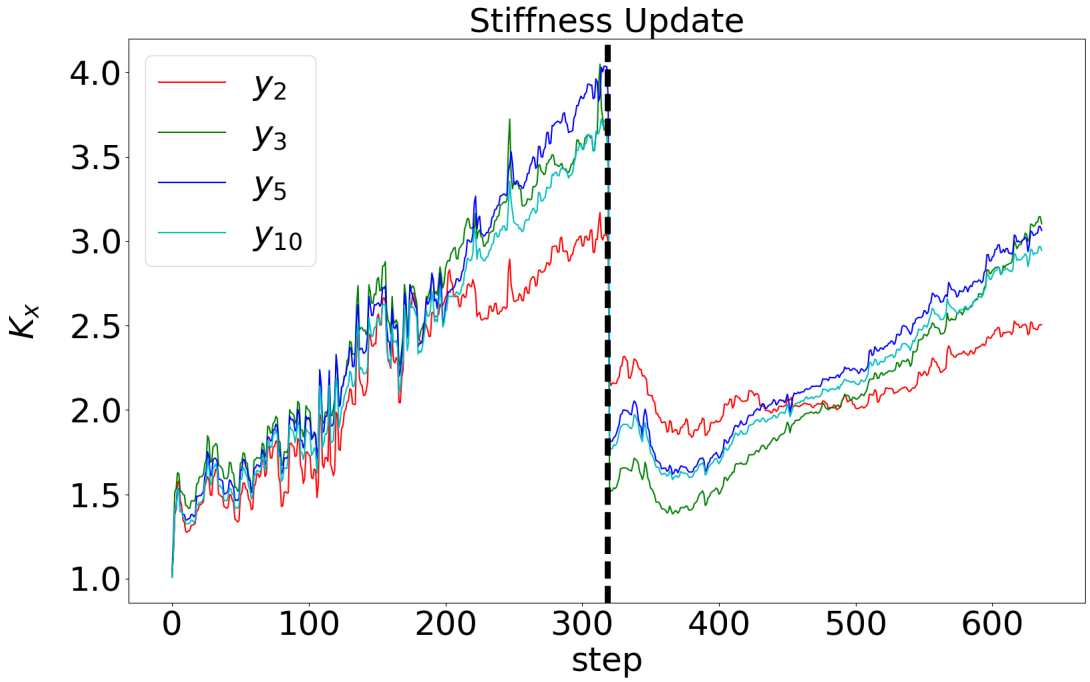


Figure 4.3: Evolution of stiffness updates as a function of applied deformations at each step for each distribution (y_2, y_3, y_5, and y_{10}), starting from an arbitrary initial value of $K_x = 1\ N.m^{-1}$.

$2\sqrt{m_1 k_{max}} \leq D_v \leq \frac{\|\dot{\boldsymbol{x}}_i \frac{m_1}{dt} + \boldsymbol{f_s}_i + \boldsymbol{f_D}_i\|}{\|\dot{\boldsymbol{x}}_i\|}$ for $\|\dot{\boldsymbol{x}}_i\| \neq 0$. When $\|\dot{\boldsymbol{x}}_i\| = 0$, $D_v = 0$ since the damping has no effect on a static point. In this work, we choose:

$$D_v = 2\sqrt{m_1 k_{max}} \tag{4.4}$$

4.1.3 Model correction by tracking

The object model is updated using (3.3), taking into account the spring forces $\boldsymbol{f_s}$ given in (3.2), which are influenced by the model parameters. However, the MSM is an approximation of the real behavior of the object. Consequently, the gap between the object model deformation and its real one will be aggregated during the object manipulation, which could lead to an unrealistic object representation. To address this issue and prevent the model from drifting from reality, a continuous tracking of the object during deformation is employed using the RGB-D camera. Whenever a deviation between the real object and its model is detected, the model is adjusted each time the camera captures a new image. To make this adjustment, we propose a modified versions of (3.1) and (3.3) already presented in 2.11:

$$m_i \ddot{\boldsymbol{x}}_i = \boldsymbol{f_s}_i + \boldsymbol{f_D}_i + \boldsymbol{f_c}_i \tag{4.5}$$

$$\boldsymbol{x}_{i(t+dt)} = \boldsymbol{x}_{i(t)} + (dt - \frac{dt^2}{m_i} D_v)\dot{\boldsymbol{x}}_{i(t)} + \frac{dt^2}{m_i}\boldsymbol{f_s}_{i(t)} + \frac{dt^2}{m_i}\boldsymbol{f_c}_{i(t)} \tag{4.6}$$

where $\boldsymbol{f_c}_i$ is a 3D force vector representing additional external constraints, which is applied on point P_i of the mesh each time a drift between the model and the real object is detected at that point.

To determine the constraint forces $\boldsymbol{f_c}$, the object is first segmented in each camera frame. Hence, by tracking $\boldsymbol{x}_o$ and $\boldsymbol{p}^o$, the drift can be detected. It appears when these two point clouds are no more superimposed. To compensate the drift, the model is adjusted by rigidly aligning $\boldsymbol{x}_o$ on $\boldsymbol{p}^o$, using an iterative closest point (ICP) procedure. As a result, $\boldsymbol{p}^o$ correspond to some points of $\boldsymbol{x}_o$ and they serve as the input displacements to $\boldsymbol{x}_o$. In [Petit, Lippiello, Fontanelli, et al., 2017], Petit et al. considered these displacements as external forces exerted by $\boldsymbol{p}^o$ on $\boldsymbol{x}_o$ and then integrated them to the mechanical model using co-rotational FEM. Similarly, using the same procedure but employing the MSM, we apply the external constraint forces $\boldsymbol{f_c}$ as follows:

$$\boldsymbol{f_c}_{o_i} = \boldsymbol{K}_{ij}(\hat{\boldsymbol{p}}^{o_i} - \boldsymbol{x}_{o_i}) \tag{4.7}$$

For the remaining mesh points not belonging to the surface, no external constraint is applied to them, *i.e.,* $\boldsymbol{f_c} = 0$. However, the effect of the force applied on the outer points of the model is propagated to the inner points with (3.3). As a result, the correction phase leads the volumetric model to approach the real object deformation behavior.

Since this correction is performed at a frequency of 30 Hz (camera frame rate), while the model is updated with a higher frequency (typically around 600 Hz), the external constraints are considered constant between two successive images.

4.1.3.1 Soft object tracking

In this paragraph, we present the tracking results achieved through the use of 4 RGB-D cameras, a soft object, and a robotic manipulator deforming the object (see Figure 4.4). Initially, the RGB-D cameras detect the object and construct its geometrical mesh, using the method elaborated in Section 4.1.1. Subsequently, the model parameters are estimated using the methodology outlined in Section 4.1.2. Once the object model is approximated using the MSM, the tracking procedure is initiated.

The objective of this experiment is to track large deformations using exclusively RGB-D data and the MSM. To facilitate deformation, a rigid ball acting as a contact tool is attached to the manipulator end-effector. However, the presence of this ball creates significant occlusion, making it impossible to track a significant part of the object surface. To address this occlusion issue, a model of the rigid ball is created, and its points are incorporated into the segmented object surface points for each camera frame. This step is straightforward, as the ball points follow the rigid movements of the robot end-effector. It is important to clarify that the displacements applied to the end-effectors do not directly impact the tracking process. Instead, these displacements are just employed to update the positions of the ball points. It is worth noting that if minor occlusion arises from the end-effector, this step is no more necessary. This holds true for the experimental results that will be presented in Section 4.2, where small sticks are attached to the robot end-effectors.

Subsequently, using correspondences between the adjusted segmented point cloud derived from the camera and the model surface points, constraint forces are computed using (4.7), and the model is iteratively refined using (4.6). To demonstrate the efficiency of the tracking procedure, captured images of the deformed object from the viewpoint of one camera are showcased in the first and third rows of Figure 4.4. Beneath each image, the corresponding resulted deformed mesh is depicted. The evolution of the mesh dur-

ing the deformation is shown in the second and fourth rows of Figure 4.4. Furthermore, for enhanced visualization of the alignment between the deformed mesh and the actual object, the mesh is also projected into the previously mentioned images. The black connected points on the soft object delineate the projected mesh. As evident from Figure 4.4, the deformed mesh accurately overlays the actual deformed object throughout the entire deformation sequence. This success underlies the effectiveness of the tracking procedure, which enables the simple MSM to faithfully reproduce the real-world deformations of the object.

Despite the simplicity of the selected model parameters, which allows high-speed real-time capacities, we will see that it is possible to consider large deformations on real objects thanks to the proposed closed-loop strategy, both in terms of control and object-model registration.

4.2 Experiments

In this section, we demonstrate the validity of the PA in real experiments involving two robotic manipulators and a RGB-D camera. We present the results of various tasks involving the indirect positioning of points on different soft objects, as well as a comparison between the PA and a model-free approach [Wada et al., 2001a].

4.2.1 Experimental setup

Our experimental setup is shown in Figure 4.5. It consists of two 6-dof robotic arms from ADEPT, a Viper 850 and a Viper 650. Each robot is equipped with a rigid stick that serves as a tool. The stick attached to the Viper 850 is dedicated solely to the deformation process throughout the experiments. On the other hand, the stick attached to the Viper 650 serves a dual purpose depending on the specified task. It prevents the object from rigidly sliding on the table when the Viper 850 is the sole robot deforming the object, and contributes in the deformation process when both robots are involved. These sticks are in contact with the object at $t = 0$ and their contact points are used as the manipulated points.

The visual perception of the object is performed with a remote RGB-D Intel Realsense-D435 camera. It acquires around 3.10^5 3D points at 30 frames per second. Prior to starting

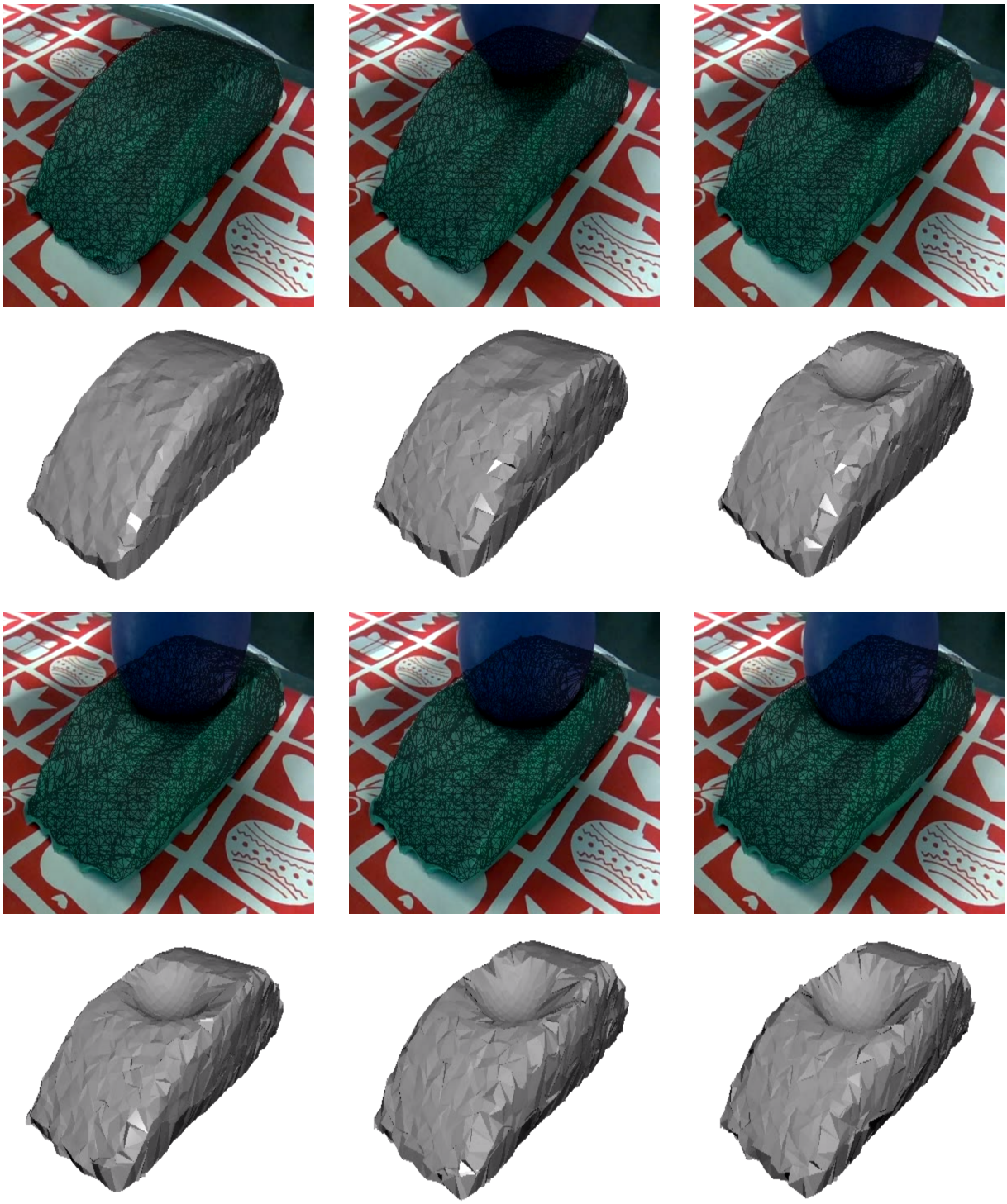


Figure 4.4: An example of the tracking results: The first and third rows show the RGB image with the projection of the object mesh (black lines) onto the object. The second and fourth rows show the deformed mesh.

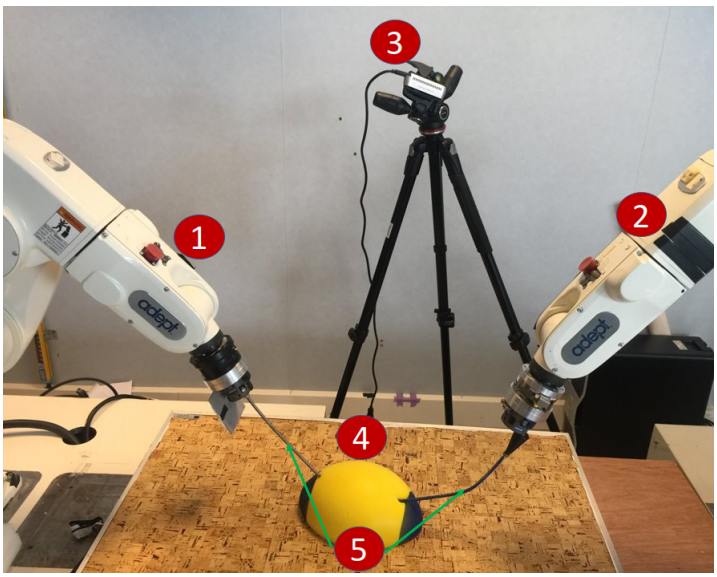


Figure 4.5: 1: Viper 650. 2: Viper 850. 3: RGB-D camera. 4: Soft objects. 5: Rigid sticks attached to Viper 850 and Viper 650.

the deformation task, the camera is utilized to create a model for each soft object used, as described in Section 4.1.1. Two types of soft objects have been considered: volumetric and planar objects, and their model parameters are determined following the procedure described in Section 4.1.2. Then, an eye-to-hand calibration is performed as the velocities generated to accomplish the deformation task are calculated in the camera frame. Since the base frame of each robot and the camera position are fixed, this calibration process only needs to be executed once, regardless of the specific task requirements. After obtaining the transformations between the camera and the robots, the velocities computed by the control scheme in the camera frame are converted and applied to the end-effectors of the robots to apply the deformations.

Moreover, the camera is also utilized to measure the position of the features points that are used as input of the control scheme, and to correct any drift between the simulated deformation using the MSM and the actual deformation of the soft object. The drift occurs when the model surface points and the real object surface points are not superimposed. In this case, external constraints are applied to the model surface points to rectify it, following the methodology explained in Section 4.1.3.

4.2.2 Experimental results

For the experimental tests, we consider either one feature point or two feature points to be positioned at some desired locations. When one feature point is considered, either one or both robots deform the soft objects simultaneously. However, when two feature points are considered, both robots are engaged. The selection of the desired position for each feature point must be carefully chosen. In other words, the desired position for each feature point should be attainable from the initial configuration of the robot(s) without damaging the object. Therefore, for each following experiment, once one or both robots come into contact with the object, we apply arbitrary movements to the robot(s), and the resulting position of each feature point is considered to be the desired position. We then repeat the experiment when the object is not deformed, aiming to achieve the same attained position. By doing so, we can ensure that the desired task is achievable without damaging the object.

As for the simulation study, the performance of the positioning task is evaluated from the evolution of the error between the current and desired positions of the feature points. In practice, this error is measured directly using the RGB-D camera by rigidly attaching one marker per feature point. In several figures of this section, colored dots will represent the projection of the desired 3D positions of the feature points in the RGB image.

The effectiveness of the servoing task for various experiments manipulating soft objects is outlined in the subsequent sections. The positioning task is considered successful when the markers and the colored dots are aligned, indicating that the feature points have reached their targeted positions.

To be noted that the depth measurements obtained from the RGB-D camera are subject to sensor noise in a range of 1 mm. Therefore, in what follows, the convergence is considered to be reached when the error norm for each feature point is less than 1 mm. Some experimental results are available on these online videos [ICARCV-video] and [Indirect-positioning-video].

4.2.2.1 One feature point

An example of the indirect positioning of a 3D feature point on a 3D object is presented in Figure 4.6. The distance between the manipulated point and the feature point is chosen arbitrary. The object is restricted on one of its sides, using the Viper 650, to prevent its free 3D translational movement. A visual marker is rigidly attached to the object at

the location of the feature point so that it is easily tracked. This marker is also used to facilitate the validation of the transportation of the feature point to the desired position. The blue dot indicates the projection of the desired 3D position of the feature point. Figure 4.6.(a) and Figure 4.6.(b) depict respectively the initial and final states for the positioning task using our proposed approach (PA). The red, green, blue and black lines in Figure 4.7 represent respectively the evolution of the positioning error along $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ axes and the error norm when using the PA. The proposed controller drives the error norm within 1 mm, between the dashed magenta lines, with an initial error around 45 mm. We can notice the nice exponential decrease of this error, which indicates the success of the positioning task and the correct modling of the system.

(a) (b)

Figure 4.6: Indirect positioning of one feature point on a 3D object: (a) 3D soft object at its rest state. (b) 3D soft object after the deformation process. The blue dot represents the projection in the image of the 3D desired position of the feature point represented by a white marker.

Another example of the indirect positioning of a 2D feature point on a planar object is represented in Figure 4.8. The object is modeled using a 2D representation, neglecting its thickness. The same methodology as the previous experiment is applied, where the Viper 650 robot restricts free 3D translational movement. The blue dot indicates the projection in the camera frame of the desired 2D position of the feature point. The object before and after deformation is depicted in Figure 4.8(a) and Figure 4.8(b), respectively. Moreover, the evolution of the positioning error during the deformation is illustrated in Figure 4.9, showing an exponential decrease. Furthermore, the error at the end of the deformation is within the range of 1 mm, which validates the success of the positioning task.

Comparison to a simple PID One commonly employed approach for indirectly positioning a feature point was proposed in [Shibata & Hirai, 2006; Wada et al., 2001a],

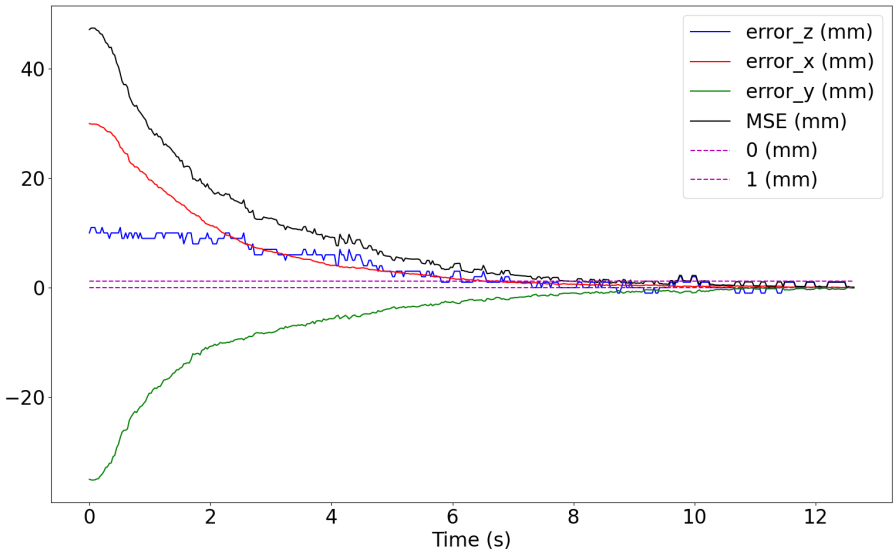
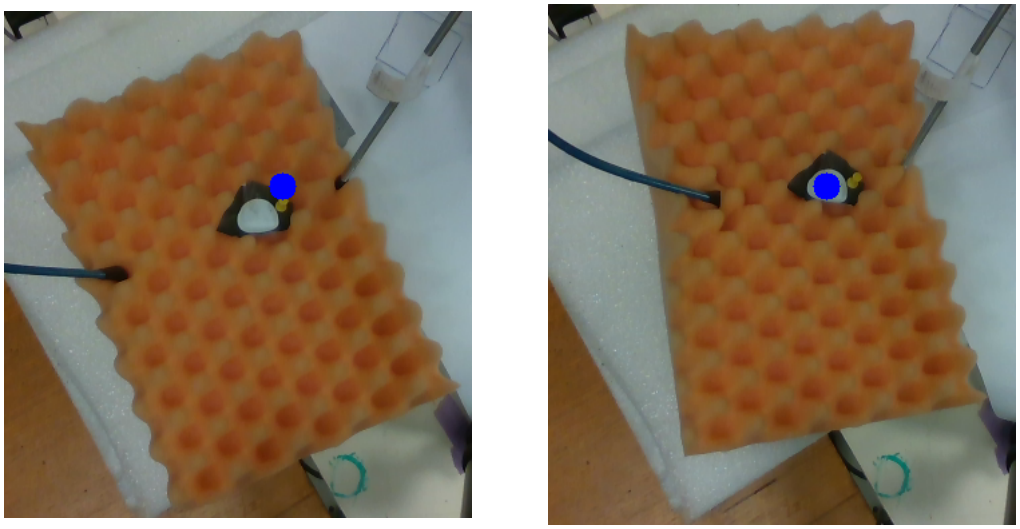


Figure 4.7: Evolution of the error for the feature point along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and the error norm of the positioning task represented in Figure 4.6.



(a) (b)

Figure 4.8: Indirect positioning of one feature point on a planar object: (a) 2D soft object at its rest state. (b) 2D soft object after the deformation process. The blue dot represents the projection in the image of the 2D desired position of the feature point represented by a white marker.

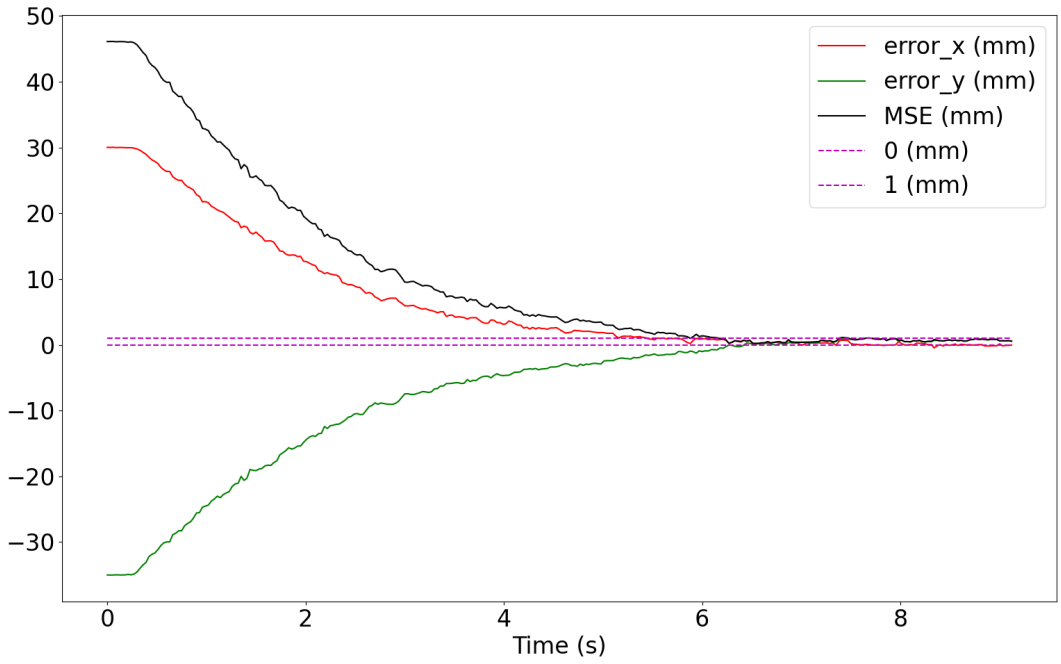


Figure 4.9: Evolution of the error for the feature point along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and the error norm of the positioning task represented in Figure 4.8.

where the authors suggested the use of a simple PID controller to accomplish the task. They define the control law as follows:

$$\dot{\boldsymbol{x}}_m = -G_p \boldsymbol{e} + G_d \dot{\boldsymbol{e}} + G_i \int \boldsymbol{e} \tag{4.8}$$

with G_p, G_d and G_i being proportional, derivative and integral gains respectively. However, while their approach is deemed valid for small deformations, they did not explicitly define what constitutes a small deformation. In our case, we call small deformation (S) when the desired position of the feature point is far from its initial position by a distance less than 10% of the largest dimension of the object, *i.e.,* height, width and thickness, and large deformation (L) when this distance is between 12% to 30% of the latter. In addition to comparing our PA with the PID controller for both small and large deformations, we perform different tests where the positions of the manipulated point and the feature one are close (N) and far apart (F). In what follows, we use some notations, such as S-N to represent small deformations and the case where the manipulated point is near to the feature point. Finally, both the proposed controller and the PID controller are tested and compared for these different types of deformations.

The main problem when using a simple PID controller is the fine-tuning of the gains of the controller for each task to be accomplished. For a small proportional gain and depending on the soft object, we noticed that the error either converges towards zero, or diverges, or converges towards a local minimum. For a large proportional gain, the error either diverges, or oscillates around a local minimum, and sometimes the object is destroyed. Therefore, we have adjusted the gains of the PID by a trial and error method. Unlike this controller, our PA does not need any parameter setting except the approximate physical model parameters and the control proportional gain, which has been fixed to $\lambda = 0.7$.

Additionally, since the robot used is equipped with a force sensor, the applied forces are measured during the deformation. The comparison results are summarized in Table 4.1. The table shows that when the manipulated point is near the feature point, the proposed method and the PID converge similarly without any problem. For the other cases, our method guarantees that the error norm converges to less than 1 mm, while it is not the case for the PID controller. The table summarizes the statistics of the different tests carried out, since many experiments have been done for each case. For example, for S-N case, many manipulated positions are chosen around the feature point within the same distance. An example of the initial state for the near case is presented in Figure 4.10. In addition, the measured forces clearly show that the robot applies less force to the object using the PA, which could be sometimes half and sometimes quarter of the applied force using the PID controller. The importance of this characteristic is that when working with some elastic objects, a high force can cause the object to loose its elasticity.

(a): S-N, error (8,15) pixels

(b): L-N, error (24,32) pixels

Figure 4.10: (a), (b) represent respectively the initial position of the feature point and its desired position for the S-N and L-N cases. The figures also present the initial absolute error in terms of pixels.

Moreover, an example of L-F case is represented in Figure 4.11. The distance between the manipulated point and the feature point is around 75 mm. The yellow rectangle

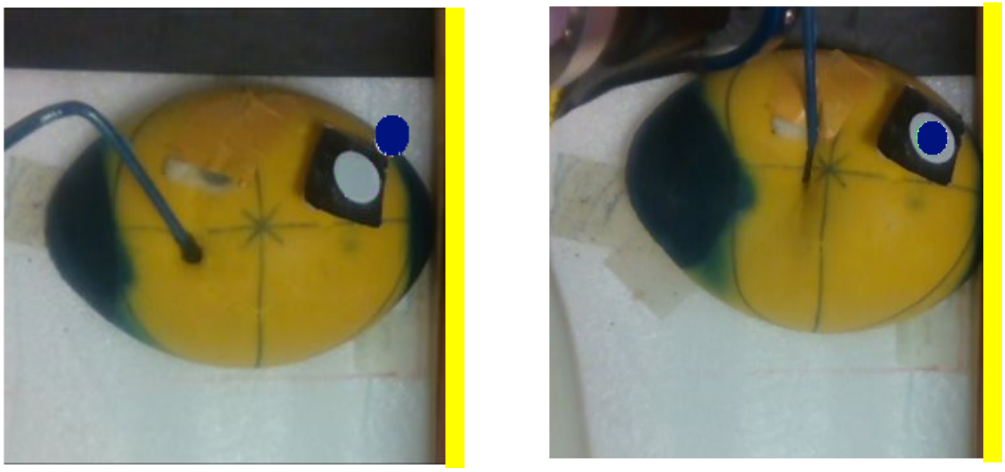
Table 4.1: PA versus a simple PID for different experiments with different initial errors, in terms of final average error $\|\bar{\boldsymbol{e}}\|$ and the maximal norm of the measured force Max $\|\boldsymbol{F}\|$ (N).

Initial error (mm)	Proposed		Simple PID	
S-N	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$
(15,10,-5)	0.7	2	0.7	4
S-F	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$
(10,10,0)	0.9	5	Diverged	> 33
L-N	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$
(35,-30,10)	0.9	10	0.9	12.25
L-F	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$	$\|\bar{\boldsymbol{e}}\|$	Max $\|\boldsymbol{F}\|_2$
(35,-30,10)	1.1	13	Diverged	> 30

presents the restricted side of the object, in order to avoid the object free 3D translation. Figure 4.11.(a) and Figure 4.11.(b) depict respectively the initial and final state for the positioning task example using our proposed approach (PA). The red, green, blue and black solid lines in Figure 4.12 represent respectively the evolution of the positioning error along $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ axes and the error norm when using the PA. The proposed controller drives the error norm within 1 mm, with an initial error around 55 mm. On the other side, the positioning error when using the simple PID controller, illustrated by dashed lines in Figure 4.12, initially decreased and then diverged.

Over-actuated case with external perturbations In this expriment, two robots are utilized to deform one feature point while external disturbances are applied to the soft object. The aim is to evaluate both the robustness of the PA against external perturbations and the positioning of the feature point when two manipulated points deform the object, rather than just one, leading to an over-actuated case (Q=1 and M=2).

The case where the feature point is in the vicinity of of its desired position is depicted in Figure 4.13. The object in its rest state is represented in Figure 4.13(a). Subsequently, the deformation process is initiated without external perturbations, except for the presence of sensor noise, and the state of the object at the end of the deformation is shown in Figure 4.13(b). Following this, a third stick is employed to manually apply a small deformation to the object in order to add a perturbation, resulting in the feature point moving from its desired position achieved previously (see Figure 4.13(c)). The system



(a) (b)

Figure 4.11: An example of L-F indirect positioning task: (a) Soft object at its rest state. (b) Soft object after the deformation process.

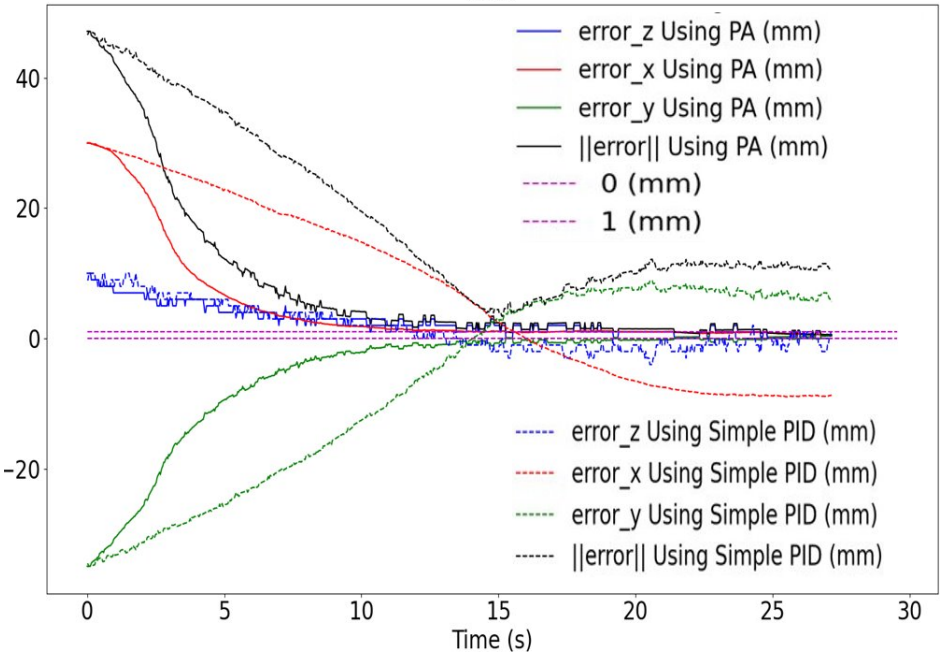


Figure 4.12: Evolution of the error for the feature point along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and the error norm of the positioning task represented in Figure 4.8. Solid lines represent the positioning error when using the PA, while dashed lines represent the positioning error when employing a PID controller.

is then allowed to re-achieve the positioning task, which is successfully accomplished as evidenced in Figure 4.13(d). In a second attempt, the same stick is used to apply a larger perturbation, as shown in Figure 4.13(e), and the same feature point is again driven to its desired position, as demonstrated in Figure 4.13(f).

Finally, the evolution of the positioning error along the three axes ($\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$) and the overall error norm of the positioning task are represented in Figure 4.14. It illustrates how the positioning error decreases exponentially to within 2 mm despite the perturbations occurring at $t = 1.0$ s and $t = 1.5$ s.

Robustness of the controller against model parameters Additional experiments are illustrated in Figure 4.15 to test the robustness of the PA against model parameter errors. The variation in error norms is presented for a L-F case, using five different controller configurations as depicted in Figure 4.15.

The error converges nicely with an exponential decrease when our PA and the identified model parameters are used (see blue graph). In contrast, the red graph shows the evolution of the error norm when the PID controller is used. We can see that the error oscillates around a local-minimum then a divergence occurs, as expected from Table 4.1.

In the third configuration, the parameters are modified: stiffness is doubled, and damping is halved. This configuration is represented by the green plot in Figure 4.15. We can observe that the error norm converges to zero but with some perturbations during the exponential decrease. This is explained by the perturbations introduced in the control scheme through $\hat{\boldsymbol{A}}$. However, when the real stiffness is multiplied by 6, the system fails in a local minimum, as indicated by the black graph. Once again, this can be explained from our discussion in Section 3.1.3.1. Since the model converges, it ensures that the condition (3.72) is met. However, due to the use of very coarse parameters, the non-negligible term ($\boldsymbol{b} - \boldsymbol{A}\widehat{\boldsymbol{A}}^{+}\widehat{\boldsymbol{b}}$) causes the error to converge to $\overline{\boldsymbol{e}} \neq \boldsymbol{0}$, and its norm converges to $|\overline{\boldsymbol{e}}| \neq 0$. In conclusion, our controller demonstrates robustness to a considerable range of coarse model parameter variations.

Lastly, the cyan graph in Figure 4.15 presents the error norm when our PA is used with a reduced mesh resolution for the object model. In this configuration, the object mesh contains 2059 nodes, as opposed to the high mesh resolution of 7103 nodes used in previous experiments. We can notice that such a reduction induces perturbations in the system behavior, but no divergence occurs, and the error still converges to a value less than 2 mm.

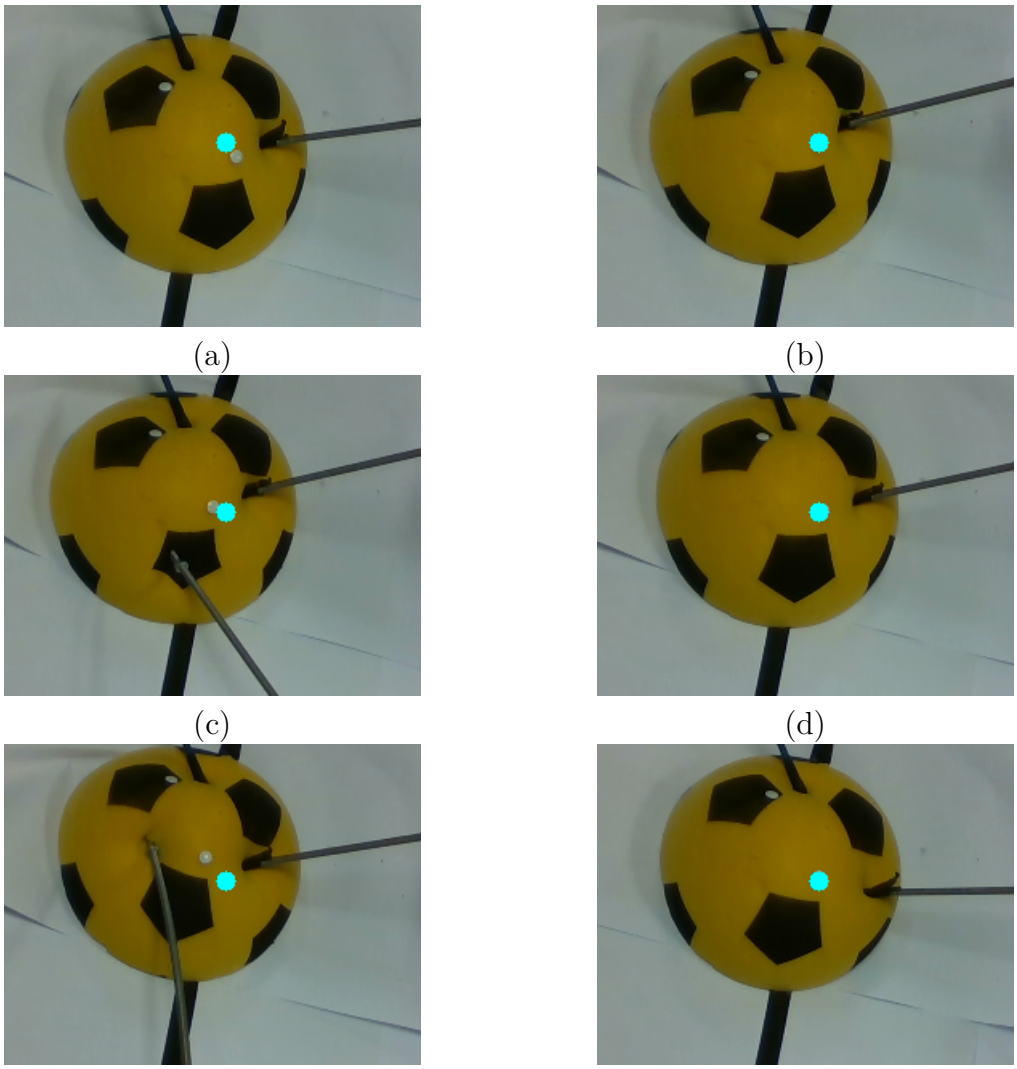


Figure 4.13: Indirect positioning of one feature point of a semi-spherical ball using two robots: (a) and (b) depict the initial and final configurations of the positioning task, respectively. (c) and (e) illustrate intermediate states of the object when it is subjected to small and large external perturbations. (d) and (f) show respectively the final states of the object once the external disturbances have been removed.

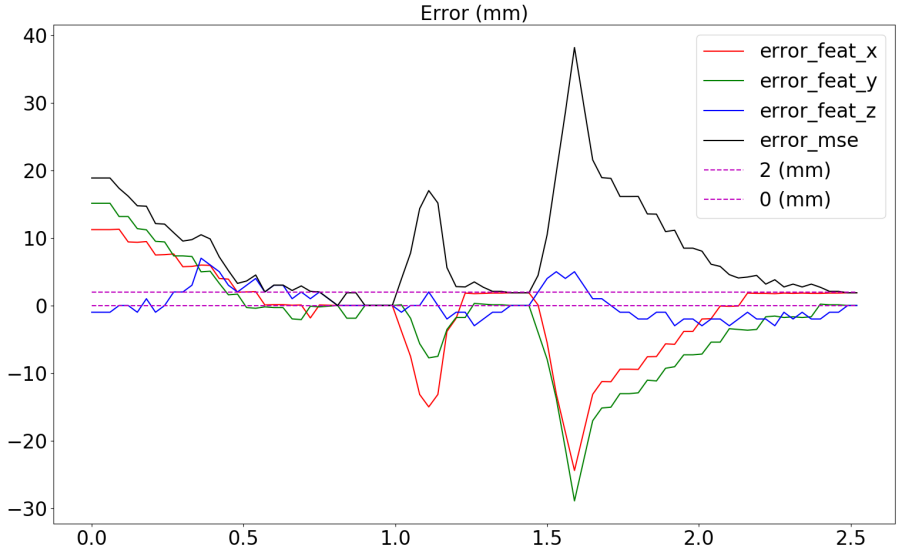


Figure 4.14: Evolution of the error for the feature point along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ and the error norm of the positioning task represented in Figure 4.13 when external perturbations are applied to the object.

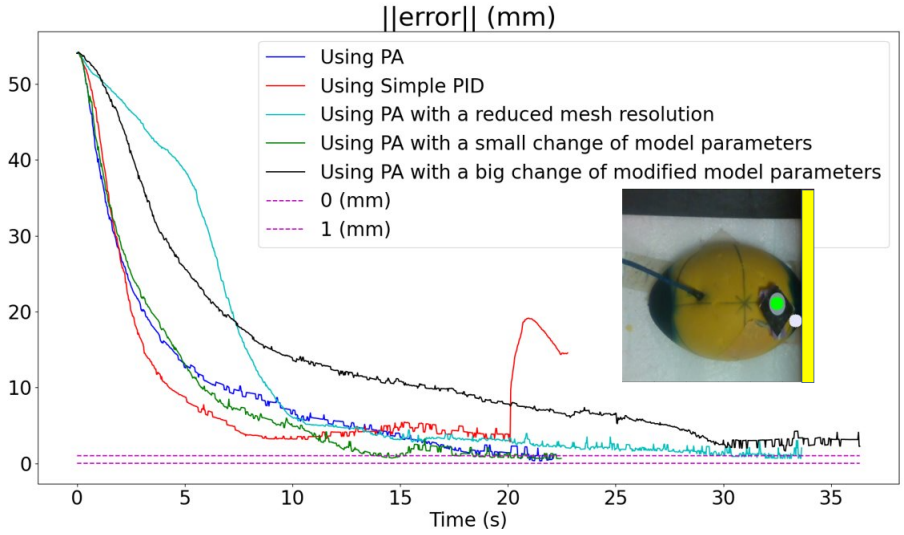


Figure 4.15: Evolution of the positioning task error norm for a L-F case (illustrated on the right of the figure) and for five different controller configurations.

4.2.2.2 Two feature points

In this section, our aim is to demonstrate the validation of the proposed control law when two feature points fixed on a soft object need to be driven to some desired positions while manipulating two robotic manipulators simultaneously. Furthermore, as shown in the previous section, the feed-forward term $\boldsymbol{b}$ distinguishes our proposed control law from the relevant model-free approaches. Hence, we also assess the significance of this term in our proposed approach. Subsequently, two different configurations are discussed in the following experiments.

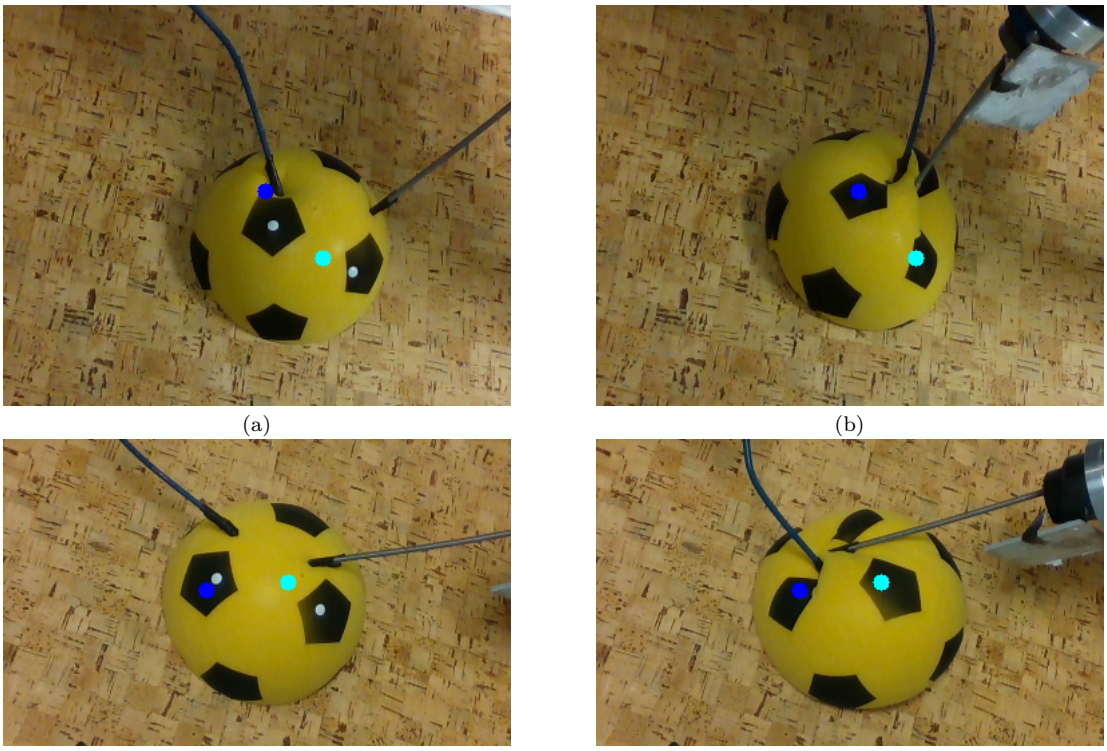
In the first configuration, two robots are employed to indirectly position two feature points on distinct soft objects in a fully actuated scenario. The purpose of this configuration is to validate the stability and efficiency of the PA for various objects with different geometries and physical properties.

In the second configuration, we focus on the examination of the impact of the feed-forward term $\boldsymbol{b}$ in (3.70).

First configuration - Fully actuated Case In this experiment, four objects with different geometries and physical parameters are considered with a fully actuated system ($M = Q = 2$).

The first object (semi-spherical ball) with two different initial placements of the manipulated points and feature point positions are presented in Figure 4.16(a) and (c). The final state of the semi-spherical ball are displayed in Figure 4.16(b) and (d) respectively. Two thumbtacks are attached to the object at the selected feature points, enabling their easy tracking through the RGB-D camera. Furthermore, in Figure 4.16, the cyan and blue dots represent the projection of the desired 3D positions of these feature points. As can be seen in Figure 4.16(b) and (d), the thumbtacks are aligned with the cyan and blue dots, indicating the successful completion of the positioning task. Furthermore, Figures 4.17 and 4.18 show the evolution of the positioning error for the tasks illustrated in Figure 4.16(a-b) and Figure 4.16(c-d) respectively. Both cases demonstrate an exponential decrease of the error norm to within 2 mm in less than 4 seconds, validating the successful positioning task.

To test the efficiency of the model, three other objects are considered: a semi-ellipsoid ball that differs from the first object in terms of geometry (see Figure 4.19), another semi-spherical ball much stiffer than the one already shown (see Figure 4.20), and finally another object with a completely different shape (see Figure 4.21). The initial and final



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

Figure 4.16: First experience for a fully actuated case using a semi-spherical ball: two configurations of positioning 2 feature points using 2 robots. (a) and (c) represent the initial state of the object while (b) and (d) depict its state at the end of the deformation, respectively. The cyan and blue dots represent the projection in the image of the 3D desired positions of the two feature points represented by white thumbtacks.

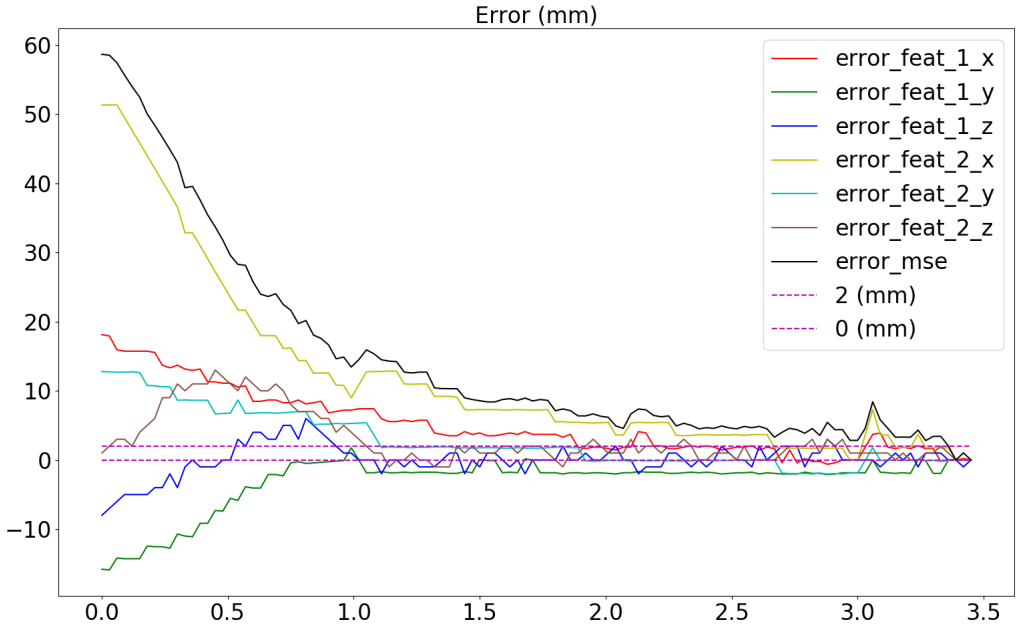


Figure 4.17: Evolution of the error for each feature point along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, and the overall error norm of the positioning task represented in Figure 4.16(a, b).

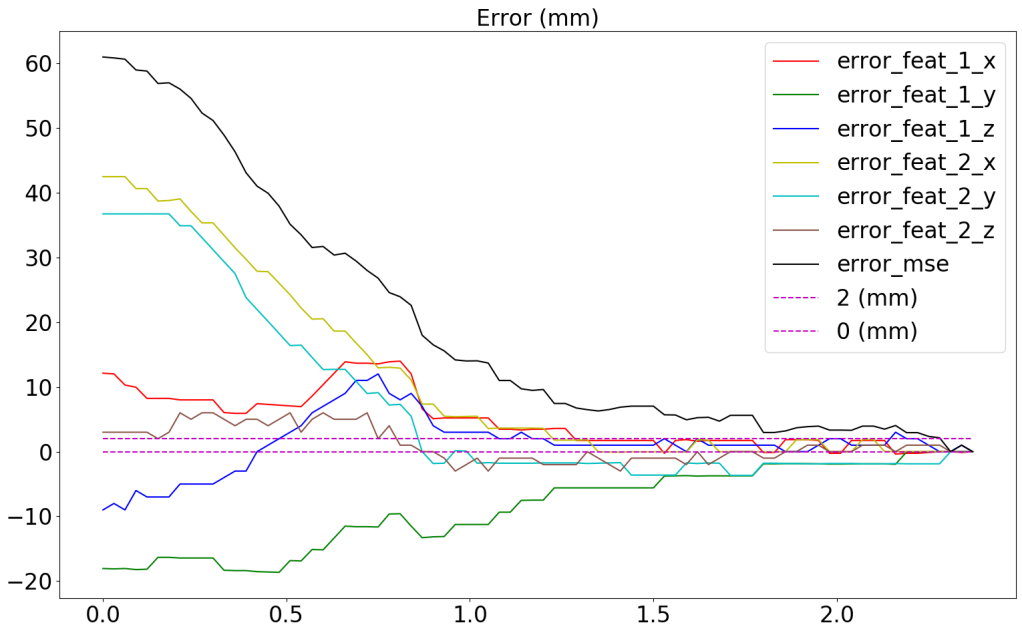
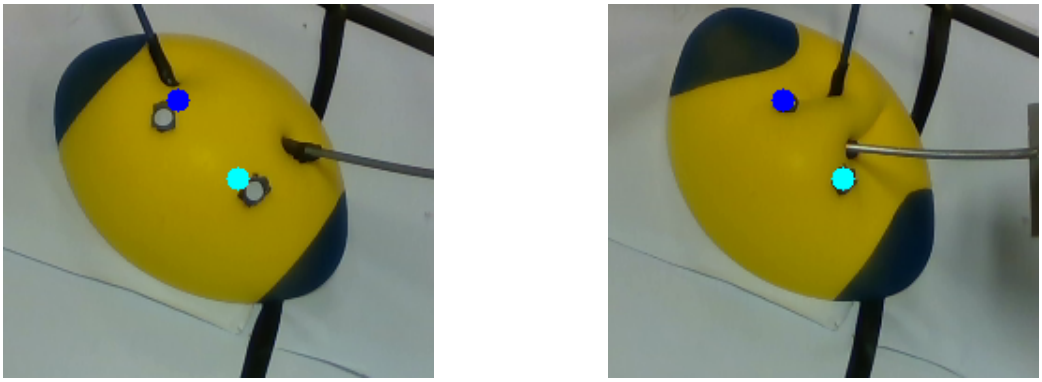


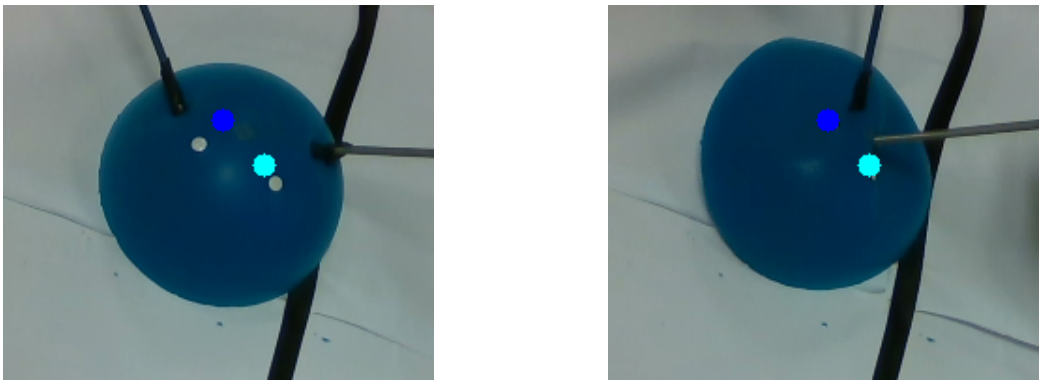
Figure 4.18: Evolution of the error for each feature point along the three axes $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, and the overall error norm of the positioning task represented in Figure 4.16(c, d).

configurations of the deformation process using each of these objects are shown in views (a) and (b) respectively. We can see on all (b) views that the markers and the blue and cyan dots are superimposed, demonstrating the success of the deformation task. Additionally, Figure 4.22 illustrates the evolution of the error norm for each of these tasks. As shown, the error norm converges to within 2 mm in less than 5 seconds, indicating the success of the positioning of the three objects.



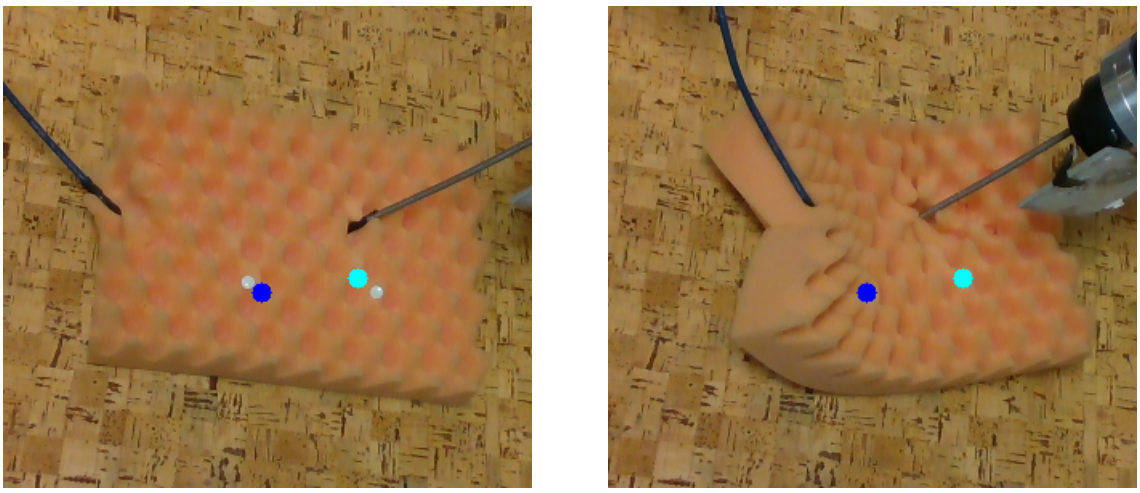
(a) (b)

Figure 4.19: Second experience for a fully actuated case using a semi-ellipsoid ball: Initial (a) and last state (b) of the object undergoing the indirect positioning of two feature points (white markers) using two robots. Colored points defined as in caption of Figure 4.16.



(a) (b)

Figure 4.20: Third experience for a fully actuated case using a semi-spherical ball: Initial (a) and last state (b) of the semi-spherical ball undergoing the indirect positioning of two feature points (white thumbtacks) using two robots. Colored points defined as in caption of Figure 4.16.



(a) (b)

Figure 4.21: Fourth experience for a fully actuated case using a planar object: (a) and (b) views and colored points defined as in caption of Figure 4.20.

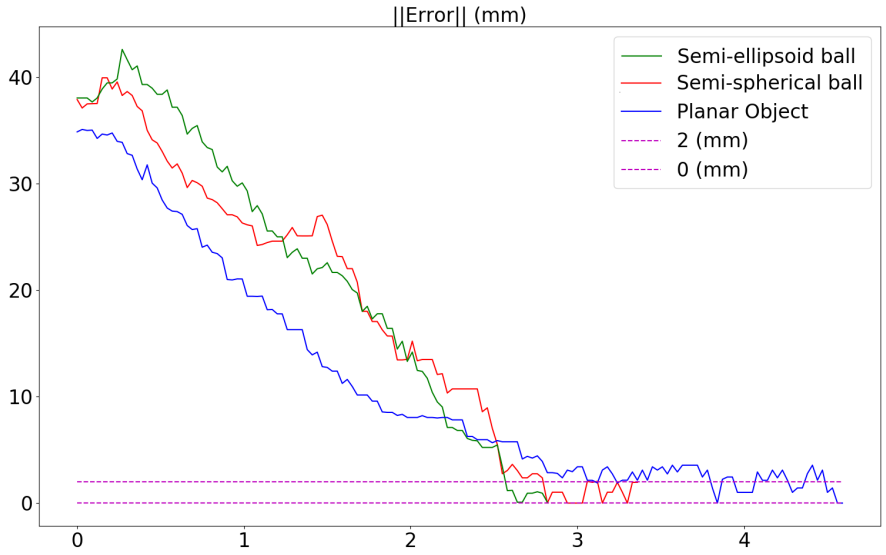
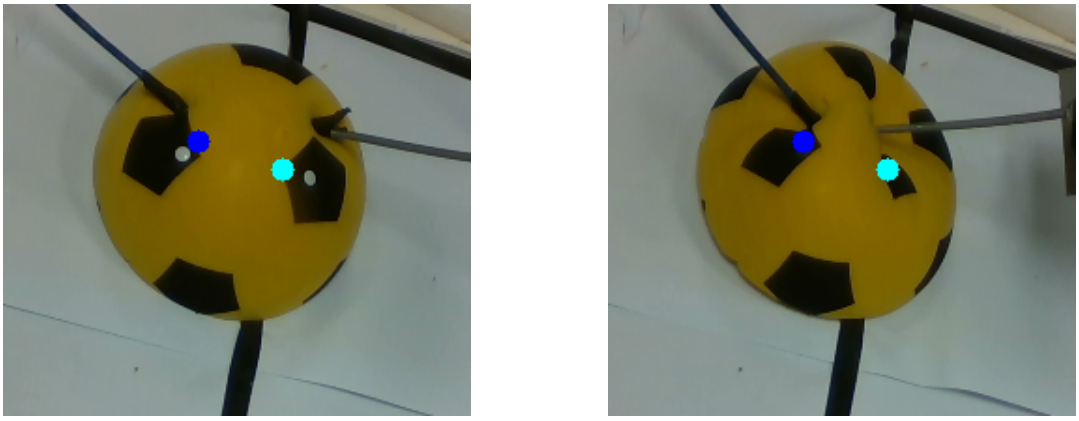


Figure 4.22: Evolution of the error norm measured at each time step for the positioning tasks, represented in Figures 4.19, 4.20 and 4.21.

Effect of the feed-forward term This paragraph presents two experiments that focus on the positioning of two feature points employing two robots. These experiments differ in whether the feed-forward term $\boldsymbol{b}$ employed in the control law (3.70) is included or not. The goal of these experiments is therefore to analyze the impact of $\boldsymbol{b}$ on the stability and successful completion of the positioning task.

The state of the object at the end of the deformation process when $\boldsymbol{b} \neq \mathbf{0}$ is shown in Figure 4.23(b). However, by setting $\boldsymbol{b} = \mathbf{0}$, the deformation task fails. The convergence and divergence for each case are illustrated in Figure 4.24.

This experiment illustrates the importance of including the feed-forward term in the control law, as it leads to an error norm of less than 2mm (red plot), while not including it leads to an oscillating and non-converging behavior (green plot), where a loss of contact with the ball occurs.



(a) (b)

Figure 4.23: Indirect positioning of 2 feature points using 2 robots for testing the impact of the feed-forward term in (3.70). Initial (a) and last state (b) of the semi-spherical ball obtained using the feed-forward term.

4.3 Conclusion

In this chapter, we validated our proposed real-time model-based and vision-based control law based on a simple MSM. We achieved this by driving various feature points on multiple soft objects with diverse geometric and physical properties to their desired positions. This manipulation was carried out by controlling one or two robotic manipulators deforming the objects simultaneously.

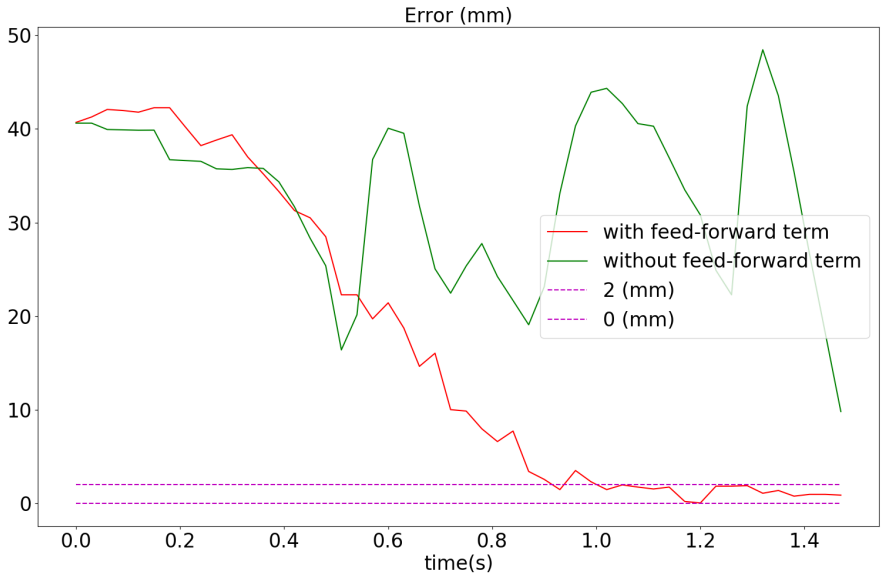


Figure 4.24: Evolution of the error norm of the positioning task, shown in Figure 4.23 with and without the feed-forward term.

In the first section, we outlined the essential steps for constructing the object model. The initial step involved the generation of a mesh that describes the object geometrical structure. Subsequently, we conducted parameter estimation for this model to establish a physical representation, enabling it to accurately replicate the real-world deformations of the actual object. These parameters are the stiffness matrix and damping coefficient of the MSM. It is important to note that these estimated parameters are approximate, potentially leading to discrepancies between the model and the real object. To address this, we introduced object tracking techniques to rectify any deviations between the model and the actual object.

In the second section, we presented experimental results obtained from interactions between one and two robot manipulators and various soft objects. These results further confirm the efficiency and robustness of the proposed control approach in dealing with external perturbations and model uncertainties. It allows to efficiently, accurately and rapidly bring multiple points of a deformable object to desired 3D positions using a simple MSM and without requiring the exact knowledge of the model parameters. To ensure that the object can be deformed so that the feature points reach their desired positions, it is however important to carefully select the positions of the manipulated points. Moreover, the attainability of the desired configuration depends on the object being deformed. For example, if the object has a limited range of motion, it may not be possible to deform it

as much as desired.

Chapter 5

Shape servoing of a soft object

In this chapter, we propose a physics-based robot controller for shape servoing of soft objects. The object shape is represented by the object 2D contour when deforming 2D objects, while it is represented by the object complete surface when dealing with 3D objects. The objective is to manipulate the soft object shape toward a desired configuration using a limited number of manipulated points. However, due to the high number of points used to represent the object shape and the limited number of manipulated points, this results in a highly under-actuated system. As a result, the targeted deformation of the system may not be ensured. For instance, achieving the deformation task would require at least one manipulated point for each point belonging to the object shape as proposed in [Wada et al., 2001a] and as discussed in Chapter 3, which is not feasible. To alleviate this problem, the shape of the deformable object is represented using a low-dimensional feature vector.

We derive two types of dimension reduction in this chapter: the first using Fourier descriptors (see Section 5.2), and the second employing 3D moments (see Section 5.3). For each of these reduction techniques, we obtain a low-dimensional vector and establish an analytical relationship describing the variation of the resulting vector in function of the manipulated points motions. As previously, these relationships are derived by approximating the object with a mass-spring model (MSM). Subsequently, we develop a control law for each shape representation using the aforementioned relations. Ultimately, we apply these developed control laws to achieve shape servoing tasks.

Both simulation and experimental results are presented, which showcase the effectiveness of our approach in soft object shape servoing.

5.1 Problem formulation

The main objective of this section is to deform a 3D soft object with w surface points, represented by

$$\boldsymbol{x}_o = (\boldsymbol{x}_{o_1}, ..., \boldsymbol{x}_{o_w}), \tag{5.1}$$

where o_i , $\forall 1 \leq i \leq w$; $o_i \in \mathcal{N}$, is the index of each surface point in $\mathcal{N}$.

Let $\boldsymbol{s}$ be a feature vector representing the 3D surface $\boldsymbol{x}_o$ of the object. The goal is to drive $\boldsymbol{s}$ to a desired target shape $\boldsymbol{s}^*$ by simultaneously manipulating M points on the object, represented by $[P_{m_1}, ..., P_{m_M}]$, with coordinates $(\boldsymbol{x}_{m_1}, ..., \boldsymbol{x}_{m_M})$.

As previously, by determining how $\boldsymbol{s}$ changes due to the motion of these manipulated points, the required motions to be applied on them for achieving the desired shape can be determined as a classical visual servoing problem [Chaumette & Hutchinson, 2008].

5.2 Shape servoing of a soft object using Fourier Descriptors

In this section, we present a new approach that controls multiple manipulator points to drive the entire shape of a soft object parametrized by a set of Fourier coefficients to a desired configuration. We employ Fourier series to describe the surface of a 3D object using spherical coordinates. In case of planar objects, Fourier series can also be used from the polar coordinates of the 2D contour shape. We use the MSM to simulate the object physical behavior and we develop the analytic relation that links the variation of the Fourier coefficients to the movements of multiple manipulated points. Based on this relation, a closed-loop control law is then developed to automatically deform the soft object toward a desired shape.

5.2.1 3D shape servoing

By expressing the object surface points $\boldsymbol{x}_{o_i}, i \in [1, ..., w]$, with spherical coordinates, the 3D cartesian position of each point can be written as:

$$\begin{cases} x_{o_i} &= x_o^g + \rho(\theta_{o_i}, \phi_{o_i}) \cos(\phi_{o_i}) \sin(\theta_{o_i}) \\ y_{o_i} &= y_o^g + \rho(\theta_{o_i}, \phi_{o_i}) \sin(\phi_{o_i}) \sin(\theta_{o_i}) \\ z_{o_i} &= z_o^g + \rho(\theta_{o_i}, \phi_{o_i}) \cos(\theta_{o_i}) \end{cases} \tag{5.2}$$

with $\boldsymbol{x}_o^g = (x_o^g, y_o^g, z_o^g)$ the 3D coordinates of the centroid of $\boldsymbol{x}_o$, $\rho(\theta_{o_i}, \phi_{o_i})$ the radial distance $\|\boldsymbol{x}_{o_i} - \boldsymbol{x}_o^g\|$ of the point P_{o_i} from $\boldsymbol{x}_o^g$, $\theta_{o_i} = \arccos(z_{o_i}/\rho(\theta_{o_i}, \phi_{o_i}))$ the polar angle and $\phi_{o_i} = \text{atan2}(y_{o_i}, x_{o_i})$ the azimuthal angle.

$\rho(\theta, \phi)$ is a closed periodic function, which can be represented as a sum of cosine and sine functions with increasing frequencies, known as Fourier descriptors. It can be approximated using Fourier series as following:

$$\begin{aligned}\rho(\theta_{o_i}, \phi_{o_i}) = \sum_{l=0}^{p}\sum_{j=0}^{q} [a_{lj}\cos(l\theta_{o_i})\cos(j\phi_{o_i}) + b_{lj}\cos(l\theta_{o_i})\sin(j\phi_{o_i}) \\ + c_{lj}\sin(l\theta_{o_i})\cos(j\phi_{o_i}) + d_{lj}\sin(l\theta_{o_i})\sin(j\phi_{o_i})]\end{aligned} \tag{5.3}$$

where a_{lj}, b_{lj}, c_{lj} and d_{lj} are the Fourier coefficients, and p and q are the number of harmonics corresponding to θ and ϕ respectively.

Hence, $\boldsymbol{s}$ representing the shape parameters is selected as:

$$\boldsymbol{s} = (x_o^g, y_o^g, z_o^g, a_{00}, ..., a_{pq}, b_{01}, ..., b_{pq}, c_{pq}, d_{pq}) \tag{5.4}$$

As noticed from (5.4), the dimension k of $\boldsymbol{s}$, $(k = 4(p+1)(q+1) - 2(p+q))$ is significantly less than the number of object surface points for low values of p and q.

Regarding the components of $\boldsymbol{s}$, they are obtained as follows:

1. The centroid of the model surface points is first computed: $\boldsymbol{x}_o^g = \frac{1}{w}\sum_{i=1}^{w}\boldsymbol{x}_{o_i}$
2. The model surface points are centered with respect to the centroid: $\hat{\boldsymbol{x}}_o = \boldsymbol{x}_o - \boldsymbol{x}_o^g$
3. The Fourier coefficients are then computed by a least squares method. In theory, only $(dim(\boldsymbol{s}) - 3)$ surface points are needed, but to increase robustness to measurement noise, all w points are considered in the computation.

Once $\boldsymbol{s}$ has been calculated, an approximation of the object surface can be reconstructed by setting the range of θ to [0, π] and ϕ to [0, 2π].

5.2.1.1 Modeling

By approximating the physical behavior of the object using the MSM, and using (3.69), we demonstrated in Chapter 3 the velocity $\dot{\boldsymbol{x}}_{o_i}$, $1 \leq i \leq w$, of any surface point due to the velocity $\dot{\boldsymbol{x}}_m = (\dot{\boldsymbol{x}}_{m_1}, ..., \dot{\boldsymbol{x}}_{m_M})$ applied on manipulated points $(P_{m_1}, P_{m_2}, ..., P_{m_M})$.

Therefore, we directly obtain the velocity of all points belonging to the model surface:

$$\dot{\boldsymbol{x}}_o = \boldsymbol{A_o}\, \dot{\boldsymbol{x}}_m + \boldsymbol{b_o} \tag{5.5}$$

with

$$\boldsymbol{A_o} = \begin{bmatrix} \boldsymbol{A}_{o_1 m_1} & \dots & \boldsymbol{A}_{o_1 m_M} \\ \boldsymbol{A}_{o_2 m_1} & \dots & \boldsymbol{A}_{o_2 m_M} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{o_w m_1} & \dots & \boldsymbol{A}_{o_w m_M} \end{bmatrix}, \boldsymbol{b_o} = \begin{bmatrix} \boldsymbol{b}_{o_1} = \sum_{l=1}^{M} \boldsymbol{b}_{o_1 m_l} \\ \vdots \\ \boldsymbol{b}_{o_w} = \sum_{l=1}^{M} \boldsymbol{b}_{o_w m_l} \end{bmatrix}, \dot{\boldsymbol{x}}_o = \begin{bmatrix} \dot{\boldsymbol{x}}_{o_1} \\ \vdots \\ \dot{\boldsymbol{x}}_{o_w} \end{bmatrix}.$$

The ultimate goal is to find the variation $\dot{\boldsymbol{s}}$ of the features with respect to the motions $\dot{\boldsymbol{x}}_m$ of the manipulated points, through the following form:

$$\dot{\boldsymbol{s}} = \boldsymbol{A_s}\, \dot{\boldsymbol{x}}_m + \boldsymbol{b_s} \tag{5.6}$$

By deriving (5.2) and (5.3), we obtain the motion $\dot{\boldsymbol{x}}_{o_i} = (\dot{x}_{o_i}, \dot{y}_{o_i}, \dot{z}_{o_i})$ of each point belonging to the object surface, as a function of $\dot{\boldsymbol{s}}$, $\dot{\theta}_{o_i}$ and $\dot{\phi}_{o_i}$. However, we have to eliminate $\dot{\theta}$ and $\dot{\phi}$ in order to express the motions of these points solely in terms of $\dot{\boldsymbol{s}}$. For that, we obtain, by linearly combining the three equations in (5.2), $\forall i \in [1, ..., w]$:

$$\alpha_{o_i}\dot{x}_{o_i} + \beta_{o_i}\dot{y}_{o_i} + \sigma_{o_i}\dot{z}_{o_i} = \left[\alpha_{o_i} \;\; \beta_{o_i} \;\; \sigma_{o_i} \;\; \left(\nu_{o_i} \frac{\partial \rho(\theta_{o_i}, \phi_{o_i})}{\partial \boldsymbol{q}}\right)\right] \dot{\boldsymbol{s}} \tag{5.7}$$

with

$$\begin{aligned} &\alpha_{o_i} = b_1 c_2 - b_2 c_1 \;\;,\;\; \beta_{o_i} = a_2 c_1 - a_1 c_2 \;\;,\;\; \sigma_{o_i} = a_1 b_2 - a_2 b_1 \\ &\nu_{o_i} = \alpha_{o_i}[(\cos(\phi_{o_i}) + \beta_{ol} \sin(\phi_{o_i})) \sin(\theta_{o_i}) + \sigma_{o_i} \cos(\theta_{o_i})] \\ &\boldsymbol{q} = (a_{00}, ..., a_{pq}, b_{01}, ..., b_{pq}, c_{pq}, d_{pq}) \end{aligned}$$

and

$$
\begin{aligned}
a_1 &= \left(\frac{\partial\rho(\theta_{o_i},\phi_{o_i})}{\partial\theta_{o_i}}\sin(\theta_{o_i}) + \rho(\theta_{o_i},\phi_{o_i})\cos(\theta_{o_i})\right)\cos(\phi_{o_i}),\\
b_1 &= \left(\frac{\partial\rho(\theta_{o_i},\phi_{o_i})}{\partial\theta_{o_i}}\sin(\theta_{o_i}) + \rho(\theta_{o_i},\phi_{o_i})\cos(\theta_{o_i})\right)\sin(\phi_{o_i}),\\
c_1 &= \frac{\partial\rho(\theta_{o_i},\phi_{o_i})}{\partial\theta_{o_i}}\cos(\theta_{o_i}) - \rho(\theta_{o_i},\phi_{o_i})\sin(\theta_{o_i}),\\
a_2 &= \left(\frac{\partial\rho(\theta_{o_i},\phi_{o_i})}{\partial\phi_{o_i}}\cos(\phi_{o_i}) - \rho(\theta_{o_i},\phi_{o_i})\sin(\phi_{o_i})\right)\sin(\theta_{o_i}),\\
b_2 &= \left(\frac{\partial\rho(\theta_{o_i},\phi_{o_i})}{\partial\phi_{o_i}}\sin(\phi_{o_i}) + \rho(\theta_{o_i},\phi_{o_i})\cos(\phi_{o_i})\right)\sin(\theta_{o_i}),\\
c_2 &= \frac{\partial\rho(\theta_{o_i},\phi_{o_i})}{\partial\phi_{o_i}}\cos(\theta_{o_i}).
\end{aligned}
$$

By using the same linear combination on (5.5) and identifying the equation obtained with (5.7), we get:

$$
\begin{bmatrix}
\alpha_{o_1}\dot{x}_{o_1} + \beta_{o_1}\dot{y}_{o_1} + \sigma_{o_1}\dot{z}_{o_1}\\
\vdots\\
\alpha_{o_w}\dot{x}_{o_w} + \beta_{o_w}\dot{y}_{o_w} + \sigma_{o_w}\dot{z}_{o_w}
\end{bmatrix}
= \boldsymbol{L_o}\,\dot{\boldsymbol{x}}_m + \boldsymbol{\delta_o} = \boldsymbol{C_o}\,\dot{\boldsymbol{s}} \tag{5.8}
$$

with

$$
\boldsymbol{L_o} = \begin{bmatrix}
 & & & & & \\
[\alpha_{o_1} & \beta_{o_1} & \sigma_{o_1}] & [\boldsymbol{A}_{o_1 m_1} & \dots & \boldsymbol{A}_{o_1 m_M}]\\
 & \vdots & & & \vdots & \\
\underbrace{[\alpha_{o_w} \quad \beta_{o_w} \quad \sigma_{o_w}]}_{1\text{x}3} & & & \underbrace{[\boldsymbol{A}_{o_w m_1} & \dots & \boldsymbol{A}_{o_w m_M}]}_{3\text{x}3M}
\end{bmatrix}
$$

$$
\boldsymbol{\delta_o} = \begin{bmatrix}
[\alpha_{o_1} & \beta_{o_1} & \sigma_{o_1}]\,\boldsymbol{b}_{o_1}\\
 & \vdots & \\
[\alpha_{o_w} & \beta_{o_w} & \sigma_{o_w}]\,\boldsymbol{b}_{o_w}
\end{bmatrix}
$$

$$
\boldsymbol{C_o} = \begin{bmatrix}
\alpha_{o_1} & \beta_{o_1} & \sigma_{o_1} & \nu_{o_1}\frac{\partial\rho(\theta_{o_1},\phi_{o_1})}{\partial\boldsymbol{q}}\\
\vdots & \vdots & \vdots & \vdots\\
\alpha_{o_w} & \beta_{o_w} & \sigma_{o_w} & \nu_{o_w}\frac{\partial\rho(\theta_{o_w},\phi_{o_w})}{\partial\boldsymbol{q}}
\end{bmatrix}
$$

By identifying (5.8) with (5.6), we finally obtain:

$$\boldsymbol{A_s} = \boldsymbol{C_o}^{+}\boldsymbol{L_o} \text{ and } \boldsymbol{b_s} = \boldsymbol{C_o}^{+}\boldsymbol{\delta_o} \tag{5.9}$$

with $\boldsymbol{C_o}^{+}$ the Moore-Penrose pseudo-inverse of $\boldsymbol{C_o}$. Matrix $\boldsymbol{C_o}$ is of dimension $w \times k$ and is full rank since the number w of surface points is highly larger than the number of harmonics. $\boldsymbol{A_s}$ can be considered as the interaction matrix by similarity with the classical visual servoing framework [Chaumette & Hutchinson, 2008], while $\boldsymbol{b_s}$ is an inertia term.

In this paragraph, we developed the variation of the low-dimensional feature vector $\boldsymbol{s}$ as a function of the manipulated point velocities, as shown in (5.6). This expression is similar to the one presented in (3.69), which was proposed for contexts involving discrete object feature point displacements in response to manipulated point motions. These equations incorporate both an interaction matrix and an inertia term, and their forms have been derived analytically.

5.2.1.2 Control scheme

This section presents the control law to be applied to the manipulated points in order to drive the object to its desired 3D shape. Similar to what was previously exploited in Section 3.1.3, which allowed us to obtain (3.70) from (3.69), the control velocities to be applied to the manipulated points to drive the visual features $\boldsymbol{s}$ to their desired values $\boldsymbol{s}^*$ are obtained from (5.6) as:

$$\dot{\boldsymbol{x}}_m = -\lambda\widehat{\boldsymbol{A}_s}^{+}\left(\boldsymbol{s} - \boldsymbol{s}^*\right) - \widehat{\boldsymbol{A}_s}^{+}\widehat{\boldsymbol{b}_s} \tag{5.10}$$

The stability analysis of the closed-loop control law proposed in (5.10) is similar to what was presented in Section 3.1.3.1. Regarding the achievement of the 3D surface deformation task, it depends on the number of harmonics p and q. When $k > 3M$, the controller (5.10) is under-actuated and cannot guarantee the asymptotic convergence of the system. Conversely, when $k \leq 3M$, the controller is either fully-actuated or over-actuated and therefore stability and convergence of $\boldsymbol{s}$ towards $\boldsymbol{s}^*$ is theoretically guaranteed as long as $\widehat{\boldsymbol{A}_s}$ is not a significantly coarse approximation of $\boldsymbol{A_s}$.

Note, however, that this does not necessarily mean that the entire object will reach its desired shape, but only its representation by the selected Fourier coefficients. Thus, the choice of the number of harmonics to be injected in $\boldsymbol{s}$ depends on the complexity of the shape to be obtained, *i.e.* complex deformations require more harmonics and

consequently more manipulated points.

5.2.2 2D shape servoing

The method described previously can easily be applied to planar objects. The shape of a 2D object is represented by its contour $\boldsymbol{x}_c$ containing w_c contour points $\boldsymbol{x}_{c_i}$. It can be parameterized by its centroid (x_c^g, y_c^g) and radial distance $\rho(\theta)$ that now only depends on a single angle θ. Doing so, any point $\boldsymbol{x}_{c_i}$ in $\boldsymbol{x}_c$ can be expressed using polar coordinates as follows:

$$\boldsymbol{x}_{c_i} = (x_c^g + \rho(\theta_{c_i})\cos(\theta_{c_i}), y_c^g + \rho(\theta_{c_i})\sin(\theta_{c_i})). \tag{5.11}$$

As $\rho(\theta)$ is periodic, it can be approximated using Fourier series as following:

$$\rho(\theta_{c_i}) = a_0^c + \sum_{l=1}^{p_c}[a_l^c\cos(l\theta_{c_i}) + b_l^c\sin(l\theta_{c_i})] \tag{5.12}$$

with p_c the number of harmonics corresponding to θ. The features $\boldsymbol{s}_c$ are thus naturally selected as:

$$\boldsymbol{s}_c = (x_c^g, y_c^g, a_0^c, a_1^c, b_1^c, ..., a_{p_c}^c, b_{p_c}^c) \tag{5.13}$$

and $\dim(\mathbf{s}_c) = (2\mathrm{p}_c + 3)$. Following the same reasoning as in Section 5.2.1.1, the variation of $\boldsymbol{s}_c$ with respect to the motions of the manipulated points $\dot{\boldsymbol{x}}_m$ can be expressed as in (5.6):

$$\dot{\boldsymbol{s}_c} = \boldsymbol{A_{s_c}}\,\dot{\boldsymbol{x}}_m + \boldsymbol{b_{s_c}} \tag{5.14}$$

with $\boldsymbol{A_{s_c}} = \boldsymbol{C_c}^{+}\boldsymbol{L_c}$ and $\boldsymbol{b_{s_c}} = \boldsymbol{C_c}^{+}\boldsymbol{\delta_c}$ where

$$\boldsymbol{L_c} = \begin{bmatrix} [\alpha_{c_1} & \beta_{c_1}][\boldsymbol{\gamma}_{c_1 m_1} & \cdots & \boldsymbol{\gamma}_{c_1 m_M}] \\ & & \vdots & \\ [\alpha_{c_{w_c}} & \beta_{c_{w_c}}][\boldsymbol{\gamma}_{c_{w_c} m_1} & \cdots & \boldsymbol{\gamma}_{c_{w_c} m_M}] \end{bmatrix},$$

$$\boldsymbol{\delta_c} = \begin{bmatrix} [\alpha_{c_1} & \beta_{c_1}]\,\boldsymbol{\delta}_{c_1} \\ & \vdots \\ [\alpha_{c_{w_c}} & \beta_{c_{w_c}}]\,\boldsymbol{\delta}_{c_{w_c}} \end{bmatrix},$$

$$\boldsymbol{C_c} = \begin{bmatrix} \alpha_{c_1} & \beta_{c_1} & \nu_{c_1}\frac{\partial\rho(\theta_{c_1})}{\partial \boldsymbol{q}_c} \\ \vdots & \vdots & \vdots \\ \alpha_{c_{w_c}} & \beta_{c_{w_c}} & \nu_{c_{w_c}}\frac{\partial\rho(\theta_{c_{w_c}})}{\partial \boldsymbol{q}_c} \end{bmatrix},$$

and

$$\begin{aligned}
\alpha_{c_i} &= \rho(\theta_{c_i})\cos(\theta_{c_i}) + \frac{\partial\rho(\theta_{c_i})}{\partial\theta_{c_i}}\sin(\theta_{c_i}),\\
\beta_{c_i} &= \rho(\theta_{c_i})\sin(\theta_{c_i}) - \frac{\partial\rho(\theta_{c_i})}{\partial\theta_{c_i}}\cos(\theta_{c_i}),\\
\nu_{c_i} &= \alpha_{c_i}(cos(\theta_{c_i}) + \beta_{c_i} sin(\theta_{c_i})),\\
\boldsymbol{q}_c &= (a_0^c, a_1^c, b_1^c, ..., a_{p_c}^c, b_{p_c}^c).
\end{aligned}$$

5.2.2.1 Control scheme

The closed-loop control law to be applied by the robotic manipulators to drive $\boldsymbol{s}_c$ to a desired contour $\boldsymbol{s}_c{}^*$ is directly deduced from (5.14):

$$\dot{\boldsymbol{x}}_m = -\lambda\widehat{\boldsymbol{A}_{\boldsymbol{s_c}}}^{+}(\boldsymbol{s}_c - \boldsymbol{s}_c{}^*) - \widehat{\boldsymbol{A}_{\boldsymbol{s_c}}}^{+}\widehat{\boldsymbol{b}_{\boldsymbol{s_c}}}, \tag{5.15}$$

with $\lambda > 0$ always denoting a positive gain and $\widehat{\boldsymbol{A}_{\boldsymbol{s_c}}}^{+}$ representing the Moore-Penrose pseudo-inverse of the approximation of $\boldsymbol{A}_{\boldsymbol{s_c}}$.

Concerning the achievement of the contour deformation task, it depends on the number of harmonics p_c. When $[dim(\boldsymbol{s}_c) = (2p_c + 3)] > 2M$, the controller in (5.15) is under-actuated and cannot guarantee the asymptotic convergence of $\boldsymbol{s}_c$ to $\boldsymbol{s}_c{}^*$. Conversely, when $dim(\boldsymbol{s}^c) < 2M$, the stability and convergence of the system is theoretically guaranteed as long as $\widehat{\boldsymbol{A}_{\boldsymbol{s_c}}}$ is not a significantly coarse approximation of $\boldsymbol{A}_{\boldsymbol{s_c}}$. As before, the choice of number of harmonics to be used depends on the complexity of the contour to be obtained, with complex deformations requiring more harmonics and hence more manipulated points.

5.2.3 Simulation results

In this section, the proposed controller is tested in simulation for deforming a 2D object to a desired shape using Fourier descriptors.

As already mentioned, the number of selected harmonics influences the accuracy of the deformation task. In other terms, it is crucial to choose an appropriate number of harmonics for obtaining a correct approximation of an object contour. For complex shapes, selecting a higher number of harmonics will improve the accuracy of the approximation. However, when the contour can be approximated using a low number of harmonics all along its deformation, there is no need to select a large number, since it will make the

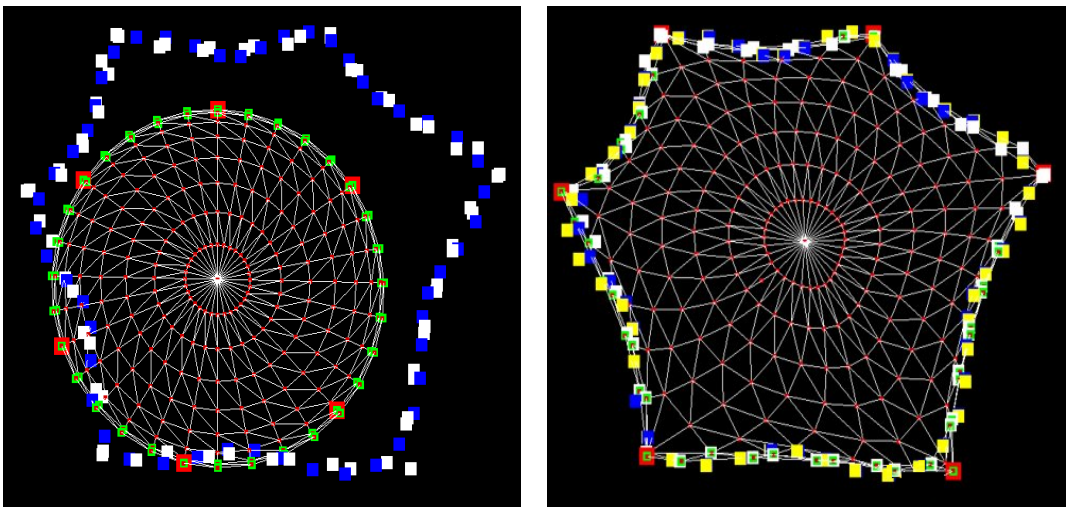
system more and more under-actuated without providing any significant information. Additionally, the desired shape must be feasible. This can be verified through applying arbitrary motions on the chosen manipulated points in order to deform the object. The resulting contour can then be selected as the desired one since we are sure that it is feasible.

The chosen 2D object is depicted in Figure 5.1. It is represented using a MSM with 256 points, where the small red points are the model points while the 6 larger red points denote the manipulated points that have been uniformly distributed around the object. Green points denote contour points ($w_c = 32$) and the white lines correspond to the springs between the model points. The white points represent the target points, which correspond to the green points acquired after manually deforming the object to obtain a feasible desired shape. The blue ones depict the desired reconstructed contour using $p_c = 8$ harmonics.

The yellow points on Figure 5.1.b correspond to the reconstructed contour at the end of the servo using the same number of 8 harmonics. We notice that the yellow points align with the green points on one side and with the blue points on the other side. The alignment with the green points indicates the successful reconstruction of the contour using 8 harmonics, while the alignment with the blue points demonstrates the accuracy and success of the shape servoing task. Furthermore, the alignment of the green points with the white points indicates the success of the contour representation using the proposed contour parameters (5.13).

The physical parameters of the object have been chosen as follows. The mass of each point is $m_i = 0.35g$. The stiffness matrix $\boldsymbol{K}_{ij}$ depends on one constant value: $\boldsymbol{K}_{ij} = diag(k_x, k_x)$ with $k_x = 13N/m$. The damping has been chosen according to Section 4.1.2.2 as $D_v = 2\sqrt{m_i k_x} = 0.13Ns/m$. Finally, we have selected $\lambda = 0.9$ as the gain for the control scheme.

Different simulations have been performed by selecting a different number of harmonics ($p_c = 6, 8$ and 10), while the number of manipulated points is always the same ($M = 6$). The purpose of testing these different numbers of harmonics is to study their effects on the deformation performance. The higher the number of harmonics, the better the reconstruction of the contour. However, from a certain amount of p_c, the reconstruction is perfect and we no longer need to increase it. To correctly reconstruct the contour of the desired shape shown in Figure 5.1, at least 6 harmonics are needed. For that, we test $p_c = 6$. Additionally, to achieve a more accurate reconstruction, we also test two



(a) (b)

Figure 5.1: Deformation of a planar object: (a) initial and desired configurations, (b) final configuration (see text for explanations on color code).

larger numbers of harmonics, $p_c = 8$ and 10. Moreover, we also aim to evaluate the achievement of the task when the system is under-actuated (here, we have 12 DOF while $dim(\boldsymbol{s}_c) = 2p_c + 3$).

The time evolution of the error norm $\|\boldsymbol{s}_c - \boldsymbol{s}_c^*\|$ for the different number of harmonics is presented in Figure 5.2. The error norm converges exponentially to nearly zero when $p_c = 6$, despite the system being under-actuated (green plot). For the other cases (blue and black plots), a very small residual remains at the convergence of the system. Recall that the same set of manipulated points is used in all these cases, and since the system is under-actuated, the rank of $\boldsymbol{A}_{\boldsymbol{s}_c}\widehat{\boldsymbol{A}_{\boldsymbol{s}_c}}^+$ is at most rank $12 < dim(\boldsymbol{s}_c)$. Consequently, following the discussion provided in Section 3.1.3.1, $\boldsymbol{A}_{\boldsymbol{s}_c}\widehat{\boldsymbol{A}_{\boldsymbol{s}_c}}^+$ has a non-zero null space, leading to a local minimum. As a result, the error norm converges to a non-zero value. Moreover, an increase in the number of harmonics results in an increase in the feature vector dimension, and as a result, the higher residual shape servoing error is observed with a greater number of harmonics.

Furthermore, the difference between the actual and desired contour is shown in Figure 5.3 for the different values of p_c. This difference has been computed using the Hausdorff distance [Taha & Hanbury, 2015] by considering all contour points. We can notice a nice exponential decrease of this error in all cases. After convergence, the remaining error is less than 3% of the initial error using 6 harmonics (green plot) while it is reduced to less

than 2% for $p_c = 8$ and 10 (blue and black plots). Since the residual of the Hausdorff distance is almost the same for two tests performed with different number of harmonics,, using 8 harmonics is here sufficient to achieve the task with high accuracy.

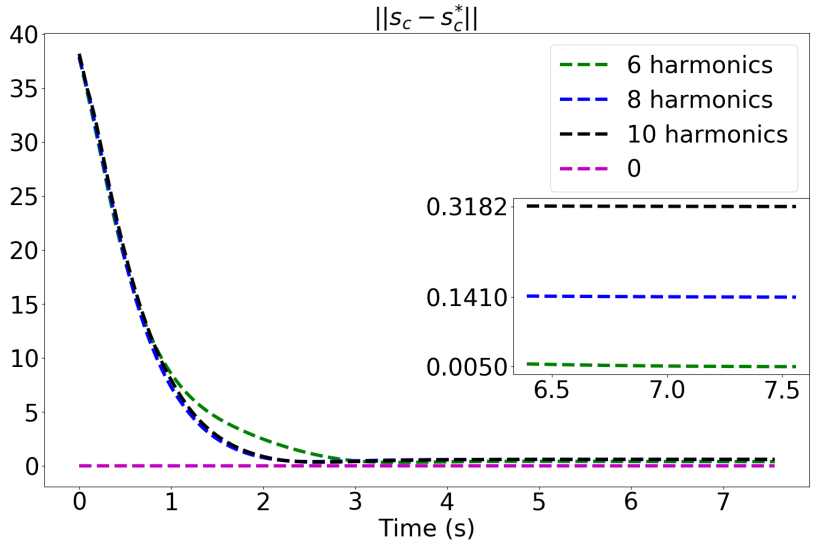


Figure 5.2: Evolution of the error norm for different numbers of harmonics and using 6 manipulated points.

5.2.4 Experimental results

Two experiments are described in this section to validate our approach. Additional experimental results can be found here: [IROS-video].

In the following experiments, we use (5.4) for the calculation of the features $\boldsymbol{s}$ and we select $p = 1$ and $q = 3$ harmonics in $\boldsymbol{s}$ since this choice enables to correctly represent the object without being excessively under-actuated. The convergence is considered achieved as soon as the error $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ reaches 5% of its initial value. For the first experiment, Figure 5.4(a) shows the object at its initial configuration while Figure 5.4(b) displays the object after convergence to its desired configuration. Figure 5.5 exhibits a 3D representation of the object surface points observed from two different viewpoints (initial configuration in red, final and desired configurations respectively in green and blue).

The second experiment involves a more complex deformation, as can be seen in Figures 5.6 and 5.7. The deformation process begins with the object in its rest state, as represented in Figure 5.6(a). Additionally, the object surface points and their desired positions are represented by green points and blue points in a 3D visualization provided

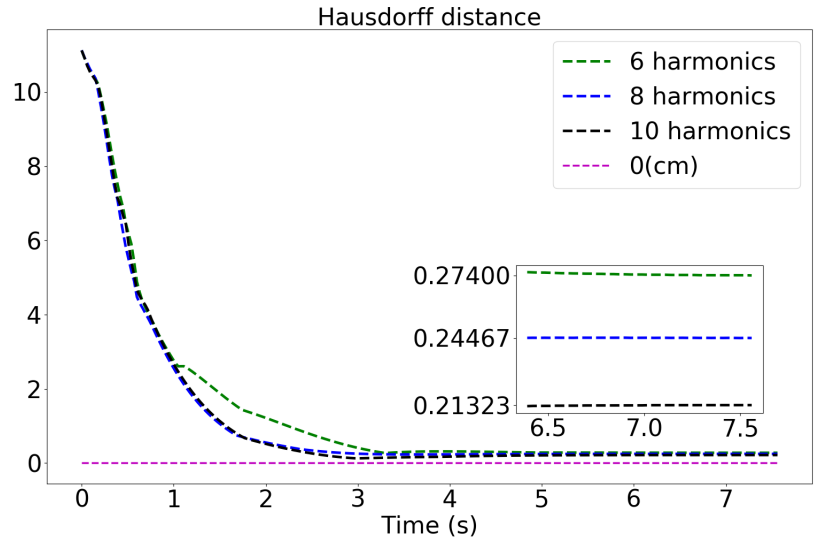
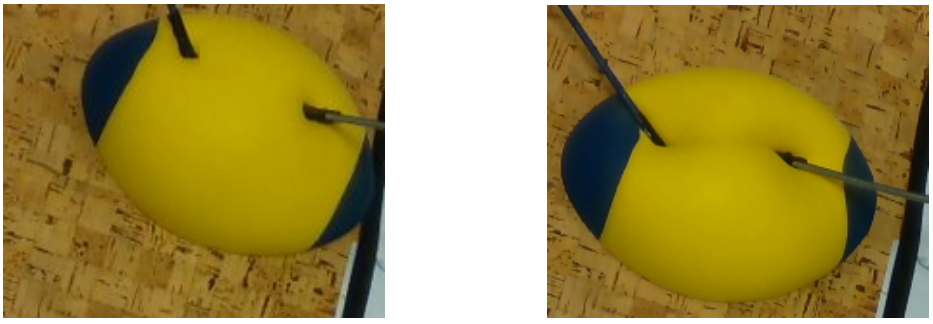
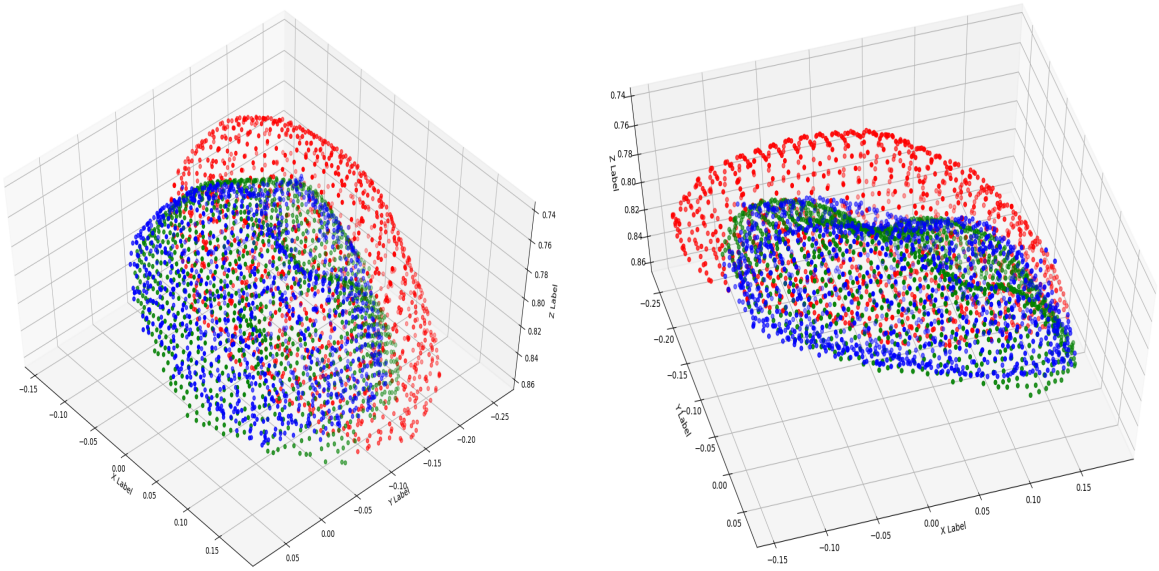


Figure 5.3: Evolution of the Hausdorff distance between the actual and desired contours for different numbers of harmonics and using 6 manipulated points.



(a) (b)

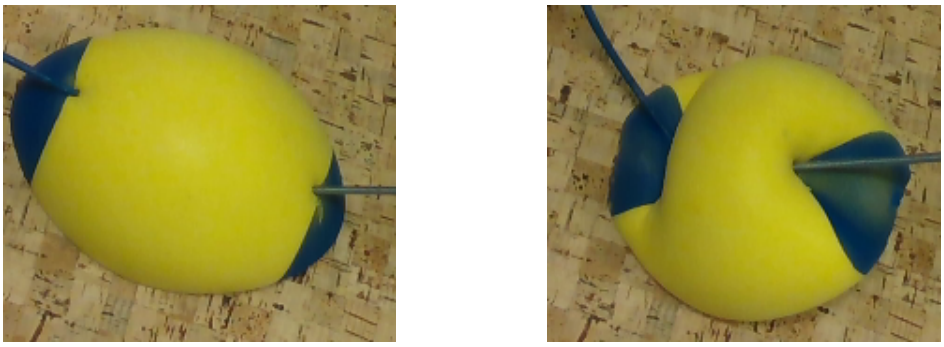
Figure 5.4: First experiment: initial (a) and final (b) images acquired by the RGB-D camera.



(a) (b)

Figure 5.5: First experiment: 3D visualization from 2 different viewpoints of the object surface points (initial positions in red, final positions in green, and desired positions in blue).

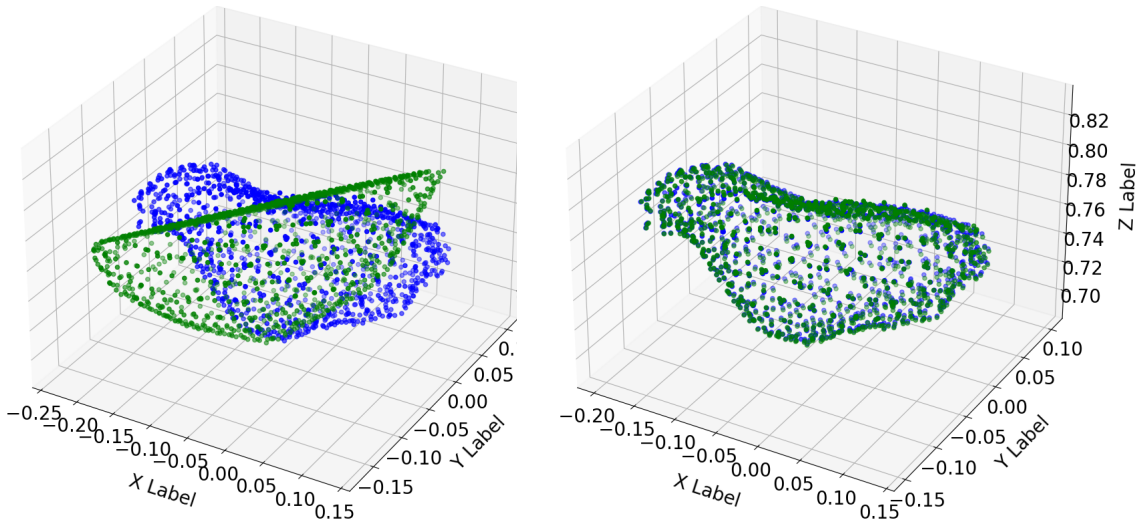
in Figure 5.7(a). Then, the shape servoing task is launched and the deformed object is depicted in Figure 5.6(b). As can be seen from Figure 5.7(b), the green points are superimposed with the blue points, indicating the success of the deformation task.



(a) (b)

Figure 5.6: Second experiment: initial and final images acquired by the RGB-D camera.

For both experiments, the time evolution of the error norm $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ is shown in Figure 5.8, while Figure 5.9 presents the evolution of the Hausdorff distance between the current and desired surface points. The convergence of the error norm to a local minimum in Figure 5.8 is evident following the same methodology provided previously. The rank of



(a) (b)

Figure 5.7: 3D visualization for the second experiment.

$\boldsymbol{A_s}\widehat{\boldsymbol{A_s}}^+$ is $6 < dim(\boldsymbol{s}) = 24$. Consequently, $\boldsymbol{A_s}\widehat{\boldsymbol{A_s}}^+$ has a non-zero null space, leading the convergence to a local minimum.

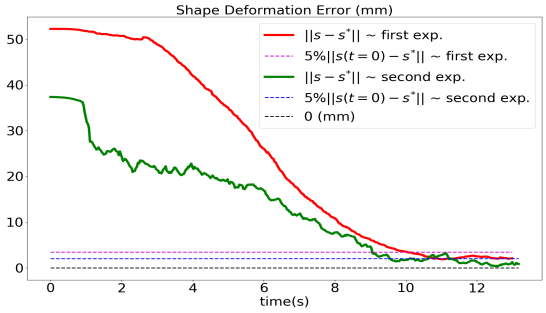


Figure 5.8: Evolution of $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ (first experiment in red, second one in green).

The correct achievement of the deformation task is directly visible from Figures 5.5 and 5.7 where we can note that the final shapes (in green) are well superposed on the desired ones (in blue) for the various deformations to be achieved, even in the presence of external perturbations. For both medium and large deformations, we can see that the features error and Hausdorff distance converge to less than 5% of their initial value in few seconds while we recall that the system is under-actuated. The decreasing is more erratic in the case of the large deformation, which is not surprising for this difficult case. Note that the initial errors are smaller in this case since the rigid displacement is smaller.

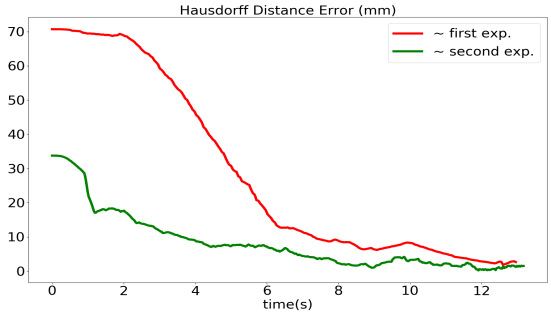


Figure 5.9: Evolution of the Hausdorff distance between the current and desired object surface points (first experiment in red, second one in green).

5.3 Shape servoing of a soft object using 3D moments

In this section, we introduce a novel method for manipulating various points on the object surface to change its shape, parameterized by a set of 3D moments, ultimately bringing it to a desired configuration.

5.3.1 3D shape servoing

This work is motivated by the work presented in [Galvez & Canton, 1993], which focused on 3D shape recognition. The 3D coordinates of the object surface points are expressed in the origin frame (center of mass) of the object. The resulted centered surface is then described using moments up to order 2, resulting in an ellipsoid representation of the object. This ellipsoid can be described by 6 parameters. Among these parameters, three correspond to the principal axes of the ellipsoid, which are associated with the eigenvectors of the inertia matrix $\boldsymbol{M}$. Additionally, the other three parameters represent the lengths of the principal axes of the ellipsoid and are related to the eigenvalues of $\boldsymbol{M}$.

The inertia matrix $\boldsymbol{M}$ is given below:

$$\boldsymbol{M} = \begin{bmatrix} \mu_{020} + \mu_{002} & -\mu_{110} & -\mu_{101} \\ -\mu_{110} & \mu_{200} + \mu_{002} & -\mu_{011} \\ -\mu_{101} & -\mu_{011} & \mu_{200} + \mu_{020} \end{bmatrix} \tag{5.16}$$

with μ_{ijk} being the central moments of order $i+j+k$ corresponding to the surface points,

which are defined as follows:

$$\mu_{ijk} = \sum_{l=1}^{w} (x_{o_l} - x_g)^i (y_{o_l} - y_g)^j (z_{o_l} - z_g)^k \tag{5.17}$$

where $\boldsymbol{x}_g = (x_g, y_g, z_g)$ is the centroid of the surface points. Let λ_1, λ_2, and λ_3 be the eigenvalues of $\boldsymbol{M}$. From these eigenvalues, the lengths of the principal axes can be calculated as follows:

$$l_1 = \sqrt{\lambda_1/w}\,,\ l_2 = \sqrt{\lambda_2/w}\,,\ l_3 = \sqrt{\lambda_3/w}$$

Let $\boldsymbol{v_1}$, $\boldsymbol{v_2}$, and $\boldsymbol{v_3}$ represent the corresponding eigenvectors of $\boldsymbol{M}$. The rotation matrix that represents the orientation of the principles axes is then given by:

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{v}_1(1)/\|\boldsymbol{v}_1\| & \boldsymbol{v}_2(1)/\|\boldsymbol{v}_2\| & \boldsymbol{v}_3(1)/\|\boldsymbol{v}_3\| \\ \boldsymbol{v}_1(2)/\|\boldsymbol{v}_1\| & \boldsymbol{v}_2(2)/\|\boldsymbol{v}_2\| & \boldsymbol{v}_3(2)/\|\boldsymbol{v}_3\| \\ \boldsymbol{v}_1(3)/\|\boldsymbol{v}_1\| & \boldsymbol{v}_2(3)/\|\boldsymbol{v}_2\| & \boldsymbol{v}_3(3)/\|\boldsymbol{v}_3\| \end{pmatrix} \tag{5.18}$$

with $\boldsymbol{v}_j(i)$ being the i^{th} element in the vector $\boldsymbol{v}_j$, and $\|\boldsymbol{v}_j\| = \sqrt{\boldsymbol{v}_j^T \boldsymbol{v}_j}$.

Following a similar approach, we first propose nine parameters to represent the object surface by an ellipsoid. The first three parameters pertain to the translational movement of the object and are represented by the coordinates of the centroid of the surface points, $\boldsymbol{x}_g$, given by:

$$x_g = \sum_{l=1}^{w} x_{o_l}/w = \mu_{100}/w\ ,\ y_g = \sum_{l=1}^{w} y_{o_l}/w = \mu_{010}/w\ ,\ z_g = \sum_{l=1}^{w} z_{o_l}/w = \mu_{001}/w \tag{5.19}$$

The remaining six parameters include the three lengths of the principal axes computed from the eigenvalues of the inertia matrix $\boldsymbol{M}$, which are obtained via second-order central moments, and the three angles that define the ellipsoid orientation. In this work, we are interested in calculating this orientation using a local rotation matrix, denoted as $\tilde{\boldsymbol{R}}$, which can be obtained by computing the difference between the actual rotation matrix, denoted as $\boldsymbol{R}$, and the desired one, denoted as $\boldsymbol{R}^*$, as follows: $\tilde{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{R}^{*T}$. Furthermore, we represent this local orientation matrix using $[\boldsymbol{\theta}\mathbf{U}]$ vector representation, which can be

obtained as follows:

$$[\boldsymbol{\theta}\mathbf{U}] = ([\boldsymbol{\theta}\mathbf{U}]_x, [\boldsymbol{\theta}\mathbf{U}]_y, [\boldsymbol{\theta}\mathbf{U}]_z) \tag{5.20}$$

$$\cos(\theta) = \left(\tilde{\boldsymbol{R}}(1,1) + \tilde{\boldsymbol{R}}(2,2) + \tilde{\boldsymbol{R}}(3,3) - 1.0\right)/2 \tag{5.21}$$

$$[\boldsymbol{\theta}\mathbf{U}]_x = \left(\tilde{\boldsymbol{R}}(3,2) - \tilde{\boldsymbol{R}}(2,3)\right)/2\sin(\theta) \tag{5.22}$$

$$[\boldsymbol{\theta}\mathbf{U}]_y = \left(\tilde{\boldsymbol{R}}(1,3) - \tilde{\boldsymbol{R}}(3,1)\right)/2\sin(\theta)$$

$$[\boldsymbol{\theta}\mathbf{U}]_z = \left(\tilde{\boldsymbol{R}}(2,1) - \tilde{\boldsymbol{R}}(1,2)\right)/2\sin(\theta)$$

with $\tilde{\boldsymbol{R}}(i,j)$ representing the element at the i^{th} row and j^{th} column of the matrix $\tilde{\boldsymbol{R}}$.

These nine parameters are utilized to represent the ellipsoid that best fits at each time the object surface points, starting from its initial configuration and moving towards its desired configuration. To efficiently represent the complete surface, additional moments with an order greater than two are then added. For this purpose, we use the following centered moments, which possess translational invariance:

$$r_1 = (\mu_{300} + \mu_{030} + \mu_{003})/w^2 \tag{5.23}$$

$$r_2 = (\mu_{140} + \mu_{014} + \mu_{401})/w^2 \tag{5.24}$$

Finally, the visual feature vector $\boldsymbol{s}$ that we propose to regulate by visual servoing is composed as follows:

$$\boldsymbol{s} = [x_g, y_g, z_g, r_1, r_2, l_1, l_2, l_3, [\boldsymbol{\theta}\mathbf{U}]_x, [\boldsymbol{\theta}\mathbf{U}]_y, [\boldsymbol{\theta}\mathbf{U}]_z] \tag{5.25}$$

5.3.1.1 Modeling

In the previous paragraph, we defined a low-dimensional feature vector $\boldsymbol{s}$ to represent the object surface points. In this paragraph, our objective is to analytically calculate the variation of $\boldsymbol{s}$ with respect to the manipulated point motions, *i.e.* $\dot{\boldsymbol{s}}$ as a function of $\dot{\boldsymbol{x}}_m$.

Since each element of $\boldsymbol{s}$ is a combination of various centered 3D moments, μ_{ijk}, we begin by providing the derivation of each element of $\boldsymbol{s}$ in terms of the centered 3D moments. This derivation allows us to express $\boldsymbol{s}$ as a function of a vector of centered 3D moments, denoted as $\boldsymbol{\mu}$. Subsequently, we derive $\boldsymbol{\mu}$ as a function of $\dot{\boldsymbol{x}}_m$. Afterward, we proceed to derive $\dot{\boldsymbol{s}}$ as a function of $\dot{\boldsymbol{x}}_m$.

The first five elements of $\boldsymbol{s}$ are defined as combinations of μ_{ijk}, as shown in (5.19), (5.23), and (5.24). Benefiting from $\boldsymbol{M}$ being a real and symmetric matrix, the analytical

formulations of eigenvalues and eigenvectors of $\boldsymbol{M}$ can be derived [Deledalle et al., 2017]. More precisely, we have:

$$\lambda_1 = [a+b+c-2\sqrt{x_1}cos(\phi\,/\,3)]\;/\,3 \tag{5.26}$$
$$\lambda_2 = [a+b+c+2\sqrt{x_1}cos((\phi-\pi)\,/\,3)]\;/\,3 \tag{5.27}$$
$$\lambda_3 = [a+b+c-2\sqrt{x_1}cos((\phi+\pi)\,/\,3)]\;/\,3 \tag{5.28}$$

with

$$a=\mu_{020}+\mu_{002}\,,\; b=\mu_{200}+\mu_{002}\,,\; c=\mu_{200}+\mu_{020}\,,\; d=-\mu_{110}\,,\; e=-\mu_{011}\,,\; f=-\mu_{101}\,,$$
$$x_1 = a^2+b^2+c^2-ab-ac-bc+3(d^2+f^2+e^2)\;,\; \phi = atan2(\sqrt{4x_1^3-x_2^2},x_2)$$
$$x_2 = -(2a-b-c)(2b-a-c)(2c-a-b)-54def$$
$$+\,9[(2c-a-b)d^2+(2b-a-c)f^2+(2a-b-c)e^2]$$

Regarding the eigenvectors, they are calculated as follows:

$$\boldsymbol{v}_1=\begin{pmatrix}(\lambda_1-c-em_1)/f\\ m_1\\ 1\end{pmatrix},\boldsymbol{v}_2=\begin{pmatrix}(\lambda_2-c-em_2)/f\\ m_2\\ 1\end{pmatrix},\boldsymbol{v}_3=\begin{pmatrix}(\lambda_3-c-em_3)/f\\ m_3\\ 1\end{pmatrix}, \tag{5.29}$$

with

$$m_1=\frac{d(c-\lambda_1)-ef}{f(b-\lambda_1)-de}\;,\; m_2=\frac{d(c-\lambda_2)-ef}{f(b-\lambda_2)-de}\;,\; m_3=\frac{d(c-\lambda_3)-ef}{f(b-\lambda_3)-de}$$

The $[\boldsymbol{\theta U}]$ vector in $\boldsymbol{s}$ is obtained from $[\boldsymbol{v}_1\;\boldsymbol{v}_2\;\boldsymbol{v}_3]$ using (5.20). These vectors depend on the eigenvalues and other elements, all of which solely rely on centered moments. As a result of this dependency, all the elements of the low-dimensional vector $\boldsymbol{s}$ are analytically calculated as functions of the centered moments. Finally, $\boldsymbol{s}$ can be expressed as follows:

$$\boldsymbol{s}=\boldsymbol{g}(\boldsymbol{\mu}) \tag{5.30}$$
$$\text{with}\;\; \boldsymbol{\mu}=[\mu_{100},\mu_{010},\mu_{001},\mu_{110},\mu_{101},\mu_{011},\mu_{200},\mu_{020},\mu_{002},\mu_{300},\mu_{030},\mu_{003},\mu_{140},\mu_{014},\mu_{401}]^T$$

By using the chain rule algorithm, $\dot{\boldsymbol{s}}$ can be calculated as $\dot{\boldsymbol{s}}=\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}\dot{\boldsymbol{\mu}}$. Therefore, we get:

$$\dot{\boldsymbol{s}}=\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}\boldsymbol{L}_{\boldsymbol{\mu}}\dot{\boldsymbol{x}}_m+\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}\boldsymbol{b}_{\boldsymbol{\mu}}=\boldsymbol{L}_{\boldsymbol{s}}\dot{\boldsymbol{x}}_m+\boldsymbol{b}_{\boldsymbol{s}} \tag{5.31}$$

with

$$\dot{\boldsymbol{\mu}} = \boldsymbol{L}_{\boldsymbol{\mu}} \dot{\boldsymbol{x}}_m + \boldsymbol{b}_{\boldsymbol{\mu}} \tag{5.32}$$

The complete derivation of $\boldsymbol{L}_{\boldsymbol{s}}$ and $\boldsymbol{b}_{\boldsymbol{s}}$ is given in Appendix (Section 7.1).

5.3.1.2 Control scheme

From the relation (5.31), we use the same strategy as in Section 5.2.1.2 to design the control scheme and we obtain:

$$\dot{\boldsymbol{x}}_m = -\lambda \widehat{\boldsymbol{L}_{\boldsymbol{s}}}^{+} (\boldsymbol{s} - \boldsymbol{s}^*) - \widehat{\boldsymbol{L}_{\boldsymbol{s}}}^{+} \widehat{\boldsymbol{b}_{\boldsymbol{s}}} \tag{5.33}$$

The stability analysis of the closed-loop control law introduced in (5.33) follows a similar discussion to that presented in Section 3.1.3.1. The achievement of the 3D surface deformation task depends on two primary factors. The first factor pertains to the rank of $\boldsymbol{L}_{\boldsymbol{s}}$ and to the actuation state of the system, whether it is over-actuated or under-actuated. The dimension of $\boldsymbol{s}$, as defined in (5.25), is 11, which means that at least 4 manipulated points must be engaged to ensure the transportation of $\boldsymbol{s}$ to $\boldsymbol{s}^*$. Consequently, if fewer than 4 manipulated points are used, the reduction of $\boldsymbol{s}$ to $\boldsymbol{s}^*$ is not guaranteed, thereby impeding the achievement of the shape servoing task. On the other hand, if $\boldsymbol{L}_{\boldsymbol{s}}$ and $\widehat{\boldsymbol{L}_{\boldsymbol{s}}}^{+}$ are of full rank 11, and $\widehat{\boldsymbol{L}_{\boldsymbol{s}}}$ is not too coarse, then error convergence to zero is theoretically ensured. The second factor is related to the representation of the object surface using the defined low-dimensional feature vector $\boldsymbol{s}$ (5.25), which is based on 3D moments. In essence, if the object surface can be adequately represented using these features, then driving $\boldsymbol{s}$ to $\boldsymbol{s}^*$ implies that the object surface will be transformed into its desired shape.

5.3.2 Simulation results

In this section, we test the ability of the proposed controller (5.33) to deform a semi-spherical 3D object, as illustrated in Figure 5.10(a). The objective is to drive its actual surface points, represented by green points, to their desired positions, indicated by blue points. This deformation is achieved through the manipulation of a limited number of three manipulated points, represented by big red points.

To validate the deformation process, we ensure the feasibility of the desired shape. Initially, the object is at rest state, and then it is deformed by arbitrarily moving the positions of the three manipulated points. The resulting deformed surface is selected

as the desired shape since it is attainable using the chosen manipulated points. Two different views of the deformation task are presented in Figure 5.10. In the first row (Perspective 1), green points represent actual object surface points, blue points denote their desired positions, and large red points correspond to manipulated points. The second row (Perspective 1) in Figure 5.10 displays green ellipsoids, which correspond to the ellipsoids parameterized by the 9 components of $\boldsymbol{s}$ that best fit the object surface points shown in the first row, while blue ellipsoids represent the ellipsoids that best fit the desired shape points, also shown in the first row. These ellipsoids correspond to points with the same colour represented in the images directly above them. For instance, the green ellipsoid in Figure 5.10(c) corresponds to the ellipsoid that fits the green points in Figure 5.10(a), while the blue ellipsoid in Figure 5.10(c) corresponds to the desired ellipsoid that fits the blue points in Figure 5.10(a). Rows 3 and 4 provide a second perspective view on the deformation process, analogous to rows 1 and 2, respectively.

Once the desired object shape is defined, we establish the initial and desired surface parameters as $\boldsymbol{s}$ and $\boldsymbol{s}^*$. The deformation task is then started with the goal of driving $\boldsymbol{s}$ towards $\boldsymbol{s}^*$. The resulting deformed object, obtained when the system converges, is illustrated in Figure 5.10(b) and Figure 5.10(f). These two images depict the same object state from different perspective views. In Figure 5.10(b), we can see the deformed state of the object from the same perspective as in Figure 5.10(a), where the object was in its initial state before stating the visual servoing. Similarly, we provide a second perspective in Figure 5.10(e) and Figure 5.10(f) showing respectively the objet in its initial and final shape.

We can observe that the green points approach the blue points in Figure 5.10(b) and Figure 5.10(f), signifying the success of the deformation task. Furthermore, in Figure 5.10(d) and Figure 5.10(h), the actual ellipsoid closely matches the desired ellipsoid. We further present in Figure 5.11 the evolution of $\|\boldsymbol{s} - \boldsymbol{s}^*\|$, which shows an exponential decrease of this error from an initial value of 17.5 to 0.142. By analogy with the discussion provided in Section 3.1.3.1, we can notice that $\text{rank}(\boldsymbol{L_s}\widehat{\boldsymbol{L_s}}^+) = 9 < \dim(\boldsymbol{s}) = 11$. Consequently, $\boldsymbol{L_s}\widehat{\boldsymbol{L_s}}^+$ has a non-zero null space, which explains why we obtained a non-zero value rather than a value close to zero.

Furthermore, we analyze in Figure 5.12 the variation in the Hausdorff distance throughout the deformation process. Initially, this distance increases slightly, mainly due to the initial error on the orientation parameters in $\boldsymbol{s}$. Despite this effect, the distance error subsequently converges to a value around 3 mm, starting from an initial value of 9 cm.

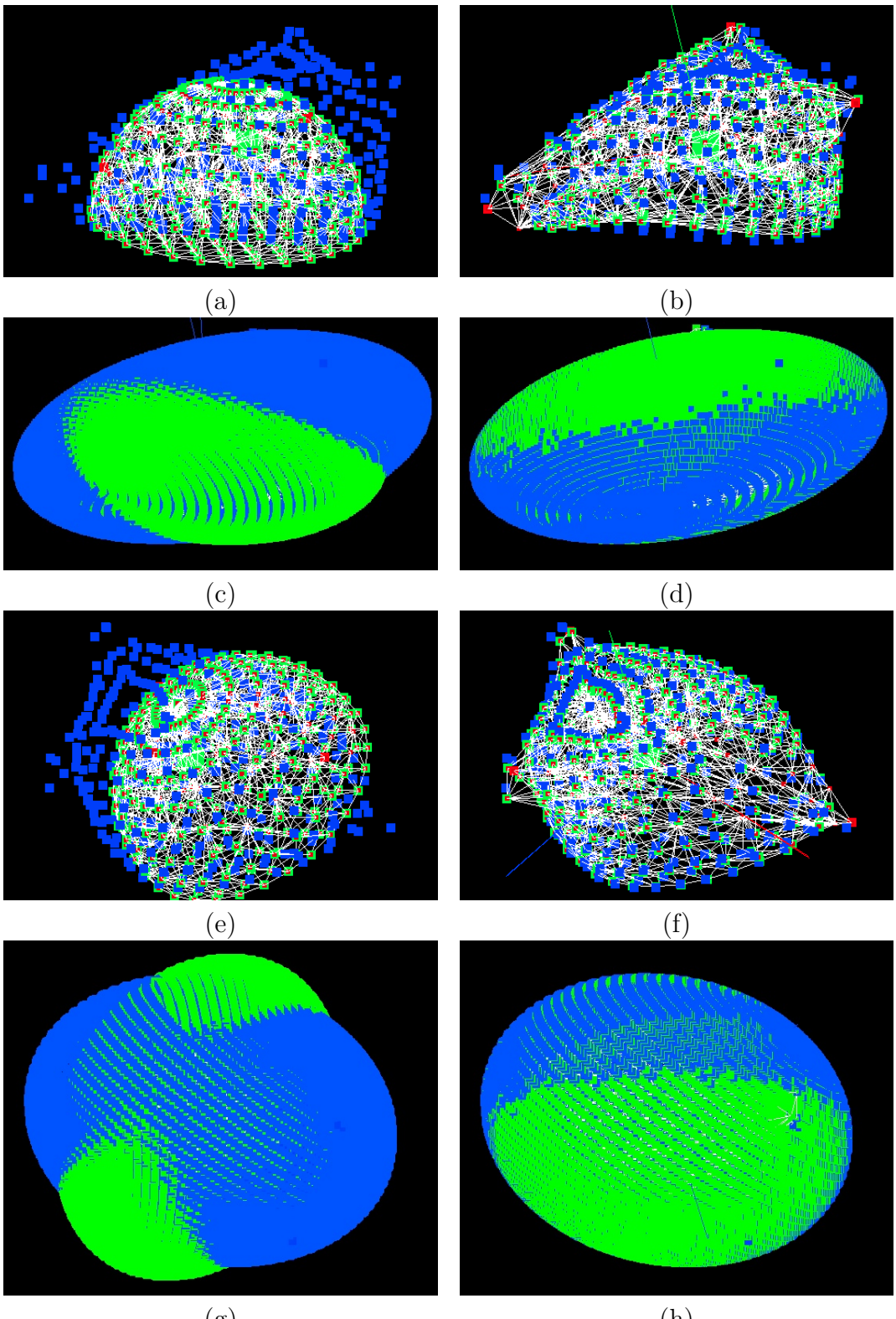


Figure 5.10: 3D Deformation of a semi-spherical ball using 3D moments: The first column depicts the initial state, while the second column illustrates the final state obtained at the end of the shape servoing task.

This low value at the convergence of the system validates the success of the deformation task, as it indicates that the actual object surface points closely approach their desired positions.

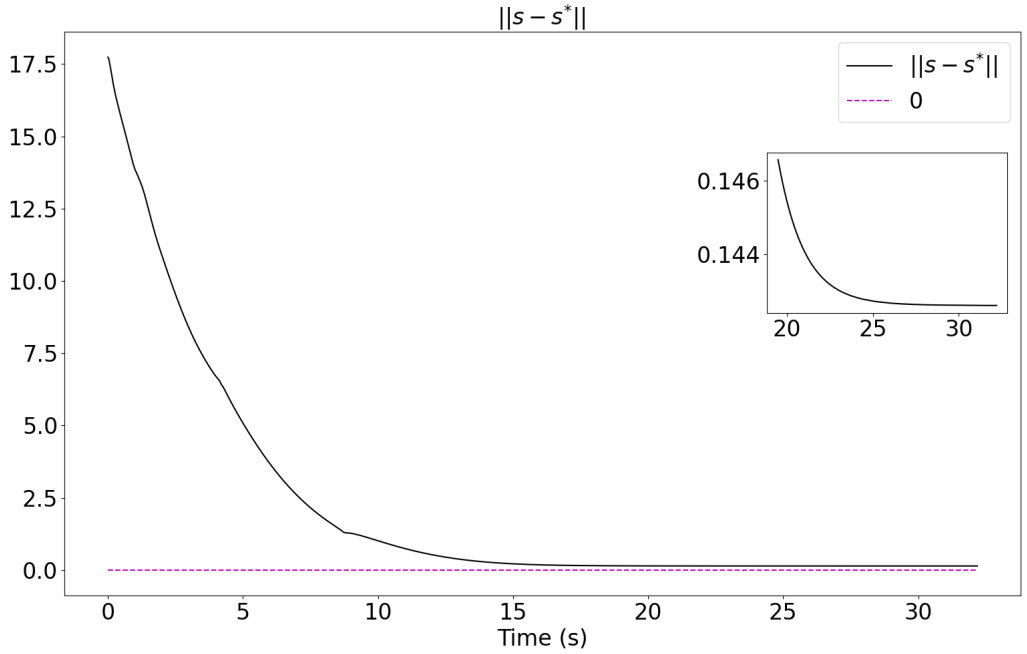


Figure 5.11: Evolution of the error norm for the shape servoing task using 3D moments and represented in Figure 5.10.

5.3.3 Experimental results

In this section, our objective is to validate the proposed approach based on 3D moments for shape servoing in an experimental scenario that includes a real soft object, two robotic manipulators, and a RGB-D camera. Since we use two robots, we can only manipulate two points, resulting in an under-actuated system ($6 < 11$). Therefore, the convergence of the error $(\boldsymbol{s} - \boldsymbol{s}^*)$ to a local minimum is expected. In such cases, we consider to achieve convergence when $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ reaches 5% of its initial value.

We start by displaying the object states before deformation (left image) and after deformation (right image) in Figure 5.13. As for our previous experiments, the sticks appearing in the images are rigid sticks attached to the robotic manipulators, with their contact points on the object serving as manipulated points. The contact between the sticks and the object is established at $t = 0$, when no deformation is applied to the object.

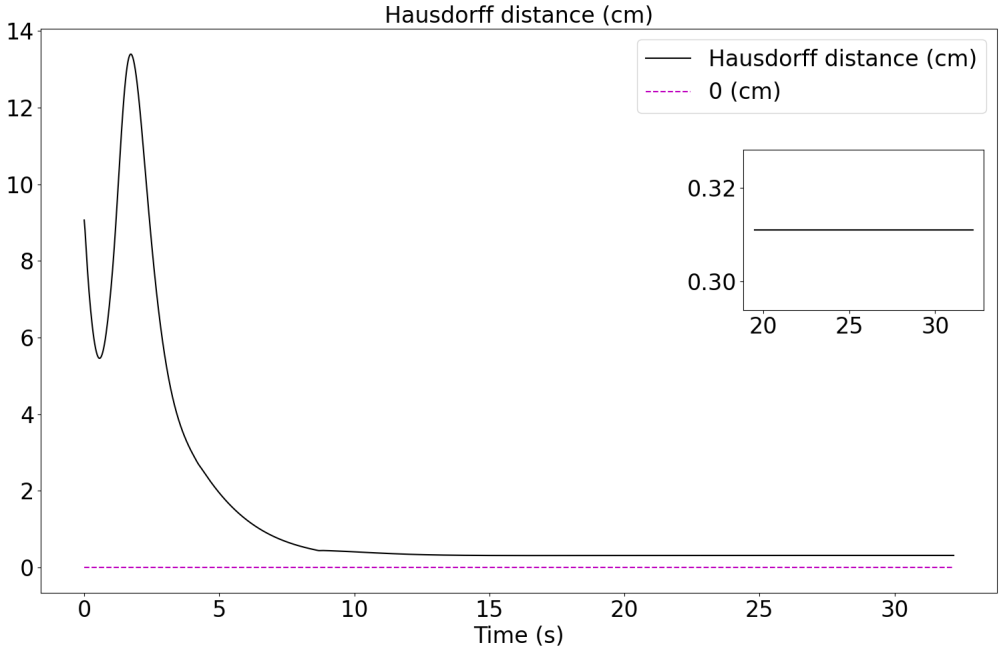


Figure 5.12: Evolution of the Hausdorff distance between the actual and desired surface points for the task represented in Figure 5.10.

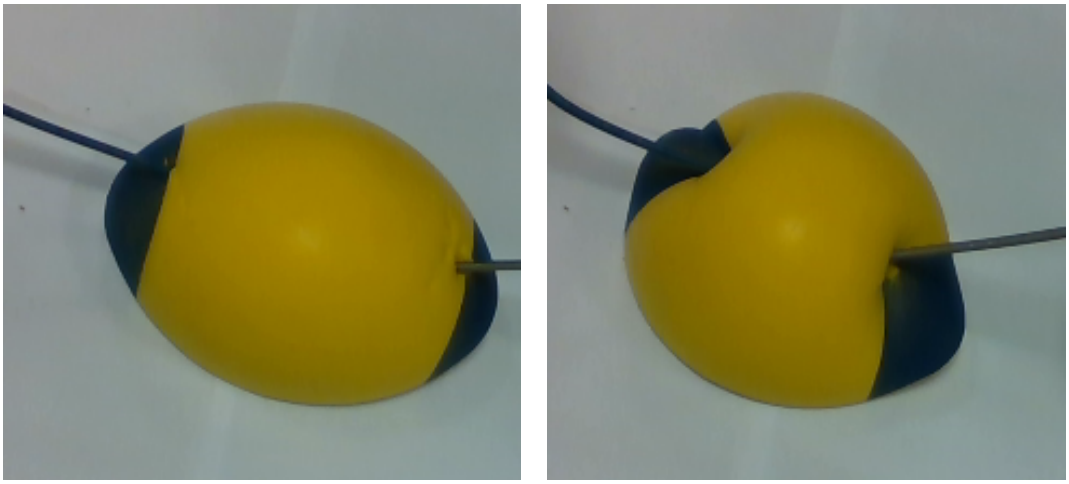
To present the deformation process more effectively, we show in Figure 5.14 the evolution of the error norm $\|\boldsymbol{s} - \boldsymbol{s}^*\|$ during the deformation task. As expected, since $\text{rank}(\boldsymbol{L_s}\widehat{\boldsymbol{L_s}}^+) = 6 < \dim(\boldsymbol{s}) = 11$, the shape servoing error norm exponentially converges to a local minimum, which is within 5% of the initial error norm.

Furthermore, to provide a clearer view of the deformation process, we visualize in Figure 5.15 the 3D surface of the object before the deformation process (red points), its final surface (green points), and its desired surface (blue points). As shown in Figure 5.15(a) and (b), the green points closely align with the blue points, indicating the successful completion of the deformation task.

5.4 Conclusion

In this chapter, we proposed two real-time model-based and vision-based control laws for driving the object shape toward a desired configuration. In other words, we validated two approaches for indirectly positioning the object surface points at desired positions by manipulating a limited number of points.

In the first approach, we approximated the object shape using Fourier descriptors,



(a) (b)

Figure 5.13: Experiment on shape servoing based on 3D moments: initial (a) and final (b) images acquired by the RGB-D camera.

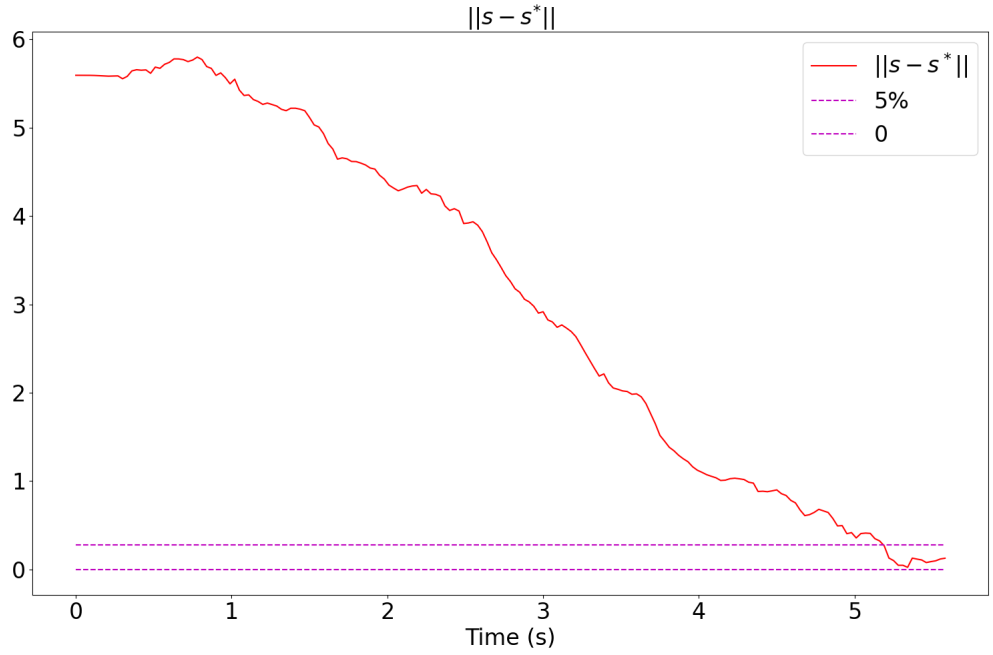
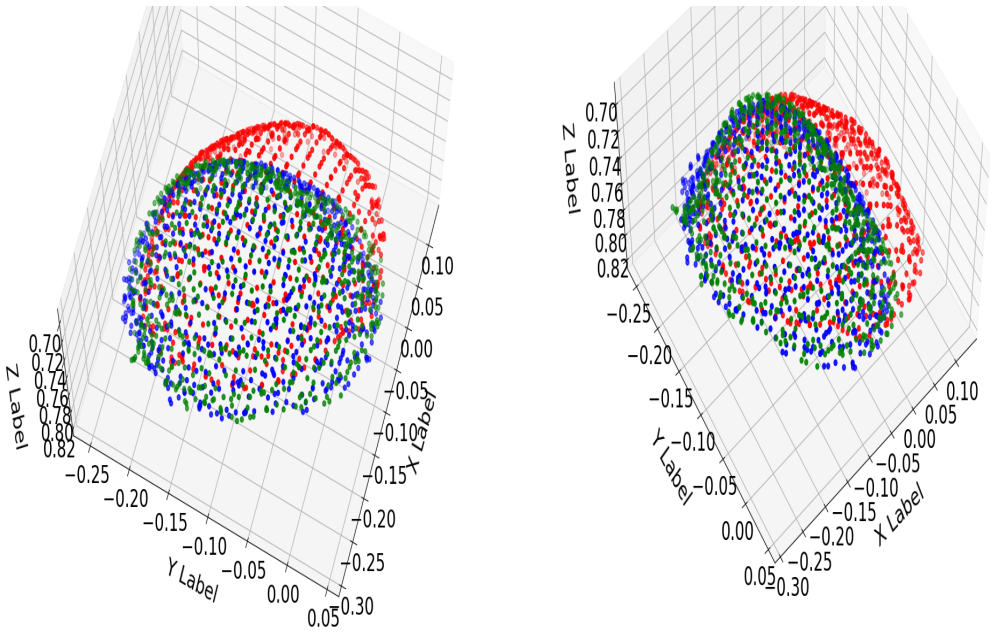


Figure 5.14: Evolution of the error norm for the experimental shape servoing task using 3D moments and represented in Figure 5.13.



(a) (b)

Figure 5.15: 3D visualization of the surface points of the object shown in Figure 5.13 from two different viewpoints (initial positions in red, final positions in green, and desired positions in blue).

which construct a low-dimensional feature vector $\boldsymbol{s}$. For a 3D object, its shape was defined by its 3D surface points, while for a 2D object, its shape was defined by its 2D contours. We then analytically derived the variation of each element in $\boldsymbol{s}$ in function of the motions to be applied to the manipulated points by approximating the soft object with a MSM. This relation constitutes the basis for our first control law for achieving shape servoing. We first validated this approach by driving the 2D contour of soft planar objects in a simulation environment. Subsequently, we validated the positioning of object surface points in real experiments, involving a real soft object and two robotic manipulators.

In the second approach, we approximated the surface of a 3D object using a low-dimensional feature vector $\boldsymbol{s}$, where its elements are obtained through a combination of 3D moments. We then analytically formulated the variation of $\boldsymbol{s}$ in response to the motions to be applied to the manipulated points, once again approximating the object using a MSM. From the equations we have developed to find this relationship, we established our second control law for shaping the object to a desired configuration. We validated this approach through simulations using synthetic soft objects, and experiments involving real soft objects with two robotic manipulators.

Chapter 6

Conclusion

6.1 Conclusion

This PhD thesis delved into into the field of robotic manipulation of soft objects, addressing various critical aspects and challenges associated with this field. The contributions and key findings from each chapter have collectively advanced our understanding and capabilities in controlling and manipulating deformable objects. Furthermore, there are numerous noteworthy applications to delve into in future work for a deeper exploration of the methodologies introduced in this thesis. These will be thoroughly examined in Section 6.2 and Section 6.3.

6.1.1 Indirect positioning of multiple points on a soft object using a simple mass-spring-model

In Chapter 3, we introduced our first contribution, which involves the analytical formulation of the dispacement of feature points belonging to a deformable object in function of the motions of manipulated points and the object model parameters. In our approach, we approximated the object using a mass-spring model, with the model stiffness matrix and damping value being its physical parameters. The success and robustness of indirectly positioning multiple object points to their desired locations were demonstrated through simulation results. In the robustness study, we addressed several key aspects, including model density (whether a sparse or dense mesh was used) and testing the manipulation task using our approach with coarse model parameters instead of exact parameters. Additionally, we demonstrated the effectiveness of the deformation task under different conditions: the positioning task error converged to zero when the system is either fully-actuated or over-actuated, and it converged to a local minimum when the system is under-actuated. Furthermore, a comparative analysis with two state-of-the-art model-free methods underscored the effectiveness of our approach in accomplishing this task,

regardless of whether the manipulated points are near or far from the feature points and whether external measurement noise is present or not.

In Chapter 4, we validated this first contribution through practical experimentations, involving real soft objects with varying geometries and material properties, two robotic manipulators, and a RGB-D camera. As an initial step, we used a methodology for reconstructing the geometric model of soft objects, enabling us to handle objects with diverse geometries. Subsequently, we presented a parameter estimation process that allows for the creation of a relative mass-spring model to represent the physical behavior of each object. Since the mass-spring model reproduces a coarse approximation of the real object behavior, there may be a drift between the model and the actual object deformations. To address this issue, we presented a correction method for this drift by tracking the real object using the RGB-D camera. This process realigns the simulation with the object actual state each time a new RGB-D image is aquired. These steps serve as a bridge from simulation results to real-world applications. Then, we validated the efficiency of our approach in indirectly positioning one or two feature points using robot arms. Furthermore, we tested the positioning task in both fully-actuated and over-actuated cases, considering the presence of external perturbations. These diverse experiments validate the applicability of our approach in real-world situations, thereby confirming the success of our first methodological contribution from both simulation and experimental results.

6.1.2 Shape servoing of a soft object using a simple mass-spring-model

In Chapter 5, we expanded upon our first contribution by introducing complete shape servoing of the soft object, which comprises the second and third contributions. The object shape is represented by its 2D contour when dealing with 2D objects and its complete 3D surface when dealing with 3D objects.

The second main contribution of this thesis involved representing the object shape using Fourier descriptors, which yields a low-dimensional feature vector. We then derived the analytical relation that links the variation of each vector element to the movements of the manipulated points. Based on this developed relations, we formulated a physics-based control law for deforming the shape of a soft object using Fourier descriptors. We successfully validated the accuracy of driving the 2D object contour to its desired contour

through simulation results. Furthermore, when dealing with 3D objects, the validation of the approach for driving the object 3D surface to its desired 3D surface was achieved through real experiments.

In the third contribution, the object shape was approximated using 3D moments. We derived analyticaly their variations in function of the displacements of the malipulated points and used this relationship in the design of the control law. In this approach, we considered only 3D objects and ensured the validity of this contribution by presenting both simulation and experimental results. In both cases, the desired shape of the object was successfully obtained.

6.2 Short-term perspectives

6.2.1 Advancing precision in deformable object manipulation - Exploring advanced control strategies and model refinement

Accurate modeling of deformable objects is crucial for achieving precision in deformation tasks. Soft objects exhibit complex behaviors that challenge traditional control approaches. Here are possible strategies for improving the models used in deformable object manipulation.

6.2.1.1 Parameter tuning

To enhance our approach, we could begin by adjusting the parameters within the mass-spring model to better match the exact properties of the deformable object in question. These parameter adjustments may encompass variables such as spring constants, mass distribution, and damping coefficients, resulting in a more precise representation of the object behavior.

While the proposed control laws were designed for a general model that allows for varying stiffness between neighboring points and different point masses, our parameter estimation process assumed a uniform stiffness matrix between adjacent points and uniform mass distribution. Therefore, a promising direction for future research would be to refine our approach to model parameter estimation to accommodate model heterogeneity.

Additionally, addressing damping within the model is of interest. Selecting appropriate damping values for the springs between adjacent points could provide a more comprehen-

sive representation of real-world scenarios. However, it is important to note that modifying the proposed control laws will be necessary to incorporate this choice of damping adjustment.

6.2.1.2 Adaptive parameterization

Another interesting future research would involve the development of algorithms capable of adaptively parameterizing the mass-spring model. These algorithms could continuously adjust the model parameters based on the object response during manipulation, thereby accommodating variations in object properties. In this scenario, object tracking during deformation would be necessary, and model parameter estimation would be employed to update the model parameters, whether choosing a simple model, as used in this thesis, or a more complex model, as proposed in the preceding paragraph.

6.2.1.3 Enhancing deformable object manipulation: Beyond PID control

In this thesis, we used a Proportional controller, but it may not always be the most suitable choice for deformable object manipulation. An alternative controller that could be suited for this task is Model Predictive Control (MPC). MPC is a control strategy that leverages a predictive model of the system to optimize control inputs over a finite time horizon.

For shape servoing of soft objects, MPC formulates the control problem as an optimization task and could rely on a mass-spring model to represent the dynamic of the object. The objective function would typically include terms like the distance between feature point positions and their desired locations or the norm of the difference between shape parameters and their desired values, depending on the desired task. The goal is to determine a sequence of control inputs, manipulating point motions, over a finite time horizon to optimize the defined objective function.

At each time step, MPC could predict the future behavior of the system by simulating the dynamic model using the current state and control inputs over the prediction horizon. This prediction would account for how the deformable object will respond to applied forces. Subsequently, MPC could optimize the control inputs over the prediction horizon to minimize the objective function. Only the first control input from the optimized sequence will be applied to the system, and this process will repeat at the next time step. This iterative approach will allow the controller to adapt to changing conditions and evolving deformable object behavior over time.

6.2.2 Complex scene - multiple objects

Another interesting future work is to implement multiple object modeling, rather than focusing on a single object. This would enable the deformation of objects within complex scenes, where the entities surrounding the object to be deformed are modeled and integrated into the deformation process.

We demonstrated the robustness of our approaches against possible external perturbations. When these perturbations originate from objects surrounding the target object, and by incorporating modeling of these objects, we could consider these perturbations in the control law. Ultimately, we would enable more precise deformations within complex scenes. The modeling procedure will not only focus on developing algorithms to simultaneously track the motion of different objects but also on modeling their interactions in a dynamic environment, including the detection and resolution of potential collisions between them.

6.3 Long-term perspectives

6.3.1 Multi-Modal sensing to enhance perception and precision in soft object manipulation

An interesting future work is to fuse tactile and vision sensors since many advantages should arise.

6.3.1.1 Enhanced perception and sensing

Dealing with vision sensors poses various challenges, particularly in positioning the sensor relative to the deformable object. For successful object parameter estimation and tracking, the deformable object has to remain within the camera field of view. However, during object manipulation, the object may become occluded by the robot grippers. To mitigate this issue, we used thin sticks (see Figure 4.5) to prevent occlusion, but this may not always be feasible. In cases like the one shown in the tracking process (see Figure 4.4), the robot end-effector can cause significant occlusion, resulting in the loss of visual information. Consequently, interesting future work would involve integrating tactile sensors to provide information on contact forces. These tactile sensors would enable the recovery of visual information lost due to occlusion by the robot grippers. For example, the estimated

applied forces could be directly integrated into the object model as constraint forces in (4.6), enabling the reproduction of the real deformation of the soft object.

6.3.1.2 Contact loss and slippage detection

During manipulation, another challenge such as contact loss between the soft object and the robotic end-effector may arise. While this contact loss cannot be easily detected using the vision sensor, tactile sensors would prove their utility in such cases. In case the tactile sensor does not register any applied force during the deformation, it could serve as an alert for contact loss. In such case, the deformation task may need to be repeated. Another solution would be to add constraints on the applied forces to prevent any contact loss.

Moreover, during manipulation, brief instances of slippage between the end-effector and the object can occur, often being unnoticed by the vision sensor. This poses a challenge for model-based approaches. Indeed, when the stick attached to the robotic manipulator makes initial contact with the object, the contact point is detected. We then establish a correspondence between the object model points and this contact point, determining the model node number, and consequently, defining the manipulated point. This correspondence remains constant, assuming that the contact point corresponds to the same model manipulated point throughout the deformation process. In cases of significant slippage, this initial correspondence must be re-evaluated, otherwise, accurate deformation may not be achieved.

In conclusion, tactile sensors provide information about contact forces and object deformation, while vision sensors offer visual information about the object shape. In future works, by integrating these two modalities, we could attain a more comprehensive understanding of the objects being manipulated while surpassing the limitations of the RGB-D sensor used.

6.3.2 Sensitive organ manipulation - Human-Machine collaboration

An interesting future surgical application would involve the delicate manipulation of sensitive organs, marking a significant advancement in medical technology and techniques. Given the sensitivity of these organs, a possible approach is to break down the deformation task into incremental steps, each requiring collaboration between a skilled medical professional, typically a surgeon, and a machine that integrates the object model and

generates a precise control law. In this collaborative process, the surgeon assumes the responsibility of both monitoring the manipulation task and intervening when necessary. Simultaneously, a machine integrates the object model and generates the control law to meet the deformation task requirements, as previously discussed in this thesis.

This paragraph delves into the intricacies of this collaborative process, shedding light on how expertise and technology could merge to achieve desired outcomes in scenarios requiring sensitive organ manipulation. Moreover, to achieve controlled and incremental deformations, especially in scenarios involving sensitive organ manipulation, this method could comprise a multi-step process. Here is a proposition of what this multi-step process might look like:

1. Optimal trajectory planning: Before initiating each deformation step, an optimal trajectory planning algorithm calculates the ideal path and motion trajectory for the robotic system or surgical tool. This planning considers the organ characteristics, constraints, and desired deformation.

2. Integration with coarse model-based control law: The results of the optimal trajectory planning are integrated into a simple model-based control law. The trajectory planning provides a set of controlled points (feature points) and intermediate desired positions. This information is then integrated into the control law, which, in turn, determines the velocities to be applied by the surgical tool to ensure that the feature points follow the planned trajectory. This step can be accomplished using the proposed approach for an indirect positioning task.

3. Real-Time monitoring: The medical professional continually monitors the deformation process in real-time, closely observing how the actual deformation aligns with the planned trajectory. If the medical professional detects significant deviations from the desired outcome or perceives potential risks to the organ integrity, he can intervene.

4. Expert corrections: Drawing on the expertise of the medical professional and sensory feedback, the physicians makes necessary adjustments to the deformation strategy to rectify any discrepancies or issues within the specific procedural goals. For example, he can halt the deformation process, correct any deviations between the output of the controlled points and their desired positions generated by the path planning, and then resume it. In the latter case, we ensure that errors do not accumulate during the process for better precision.

5. Gradual deformation progression: With each iteration, the organ undergoes a controlled deformation increment. This incremental approach continues until the overall deformation goal is achieved. The aim of these incremental iterations is to ensure that, at each step, the medical professional can detect any potential harm to the organ, allowing for timely intervention based on their deep knowledge of organ behavior and anatomical considerations.

6. Ensuring safety and precision: Throughout the entire process, the combined efforts of trajectory planning, automated control inputs, and expert intervention maintain a delicate balance between safety and precision, ultimately achieving the desired organ deformation while prioritizing patient safety and optimal medical outcomes.

This multi-step approach reflects a cautious methodology for achieving controlled deformations in sensitive organ manipulation scenarios. It leverages both computational models and human expertise to optimize the process while ensuring the highest standards of patient safety and desired medical results.

Chapter 7

Appendix

7.1 Shape servoing based on 3D moments - Derivation of the interaction matrix and the feed-forward term

In this appendix, we derive analytically the required equations to obtain $\boldsymbol{L_s}$ and $\boldsymbol{b_s}$ as defined in (5.31). We can notice that these elements depend on $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}$, $\boldsymbol{L_\mu}$ and $\boldsymbol{b_\mu}$.

We begin by calculating $\boldsymbol{L_\mu}$ and $\boldsymbol{b_\mu}$. By approximating the physical behavior of the object using the MSM, we obtain similarly to (5.5) the following equation for any surface point P_{o_i}, with $1 \leq i \leq w$:

$$\dot{x}_{o_i} = \sum_{l=1}^{M} \boldsymbol{A}_{o_i m_l}(1,:)\dot{\boldsymbol{x}}_{m_l} + \boldsymbol{b}_{o_i}(1) \tag{7.1}$$

$$\dot{y}_{o_i} = \sum_{l=1}^{M} \boldsymbol{A}_{o_i m_l}(2,:)\dot{\boldsymbol{x}}_{m_l} + \boldsymbol{b}_{o_i}(2) \tag{7.2}$$

$$\dot{z}_{o_i} = \sum_{l=1}^{M} \boldsymbol{A}_{o_i m_l}(3,:)\dot{\boldsymbol{x}}_{m_l} + \boldsymbol{b}_{o_i}(3) \tag{7.3}$$

with $\boldsymbol{A}_{o_l m_l}(i,:)$ representing the i^{th} row of the 3x3 matrix $\boldsymbol{A}_{o_l m_l}$. From the definition of μ_{ijk} given in (5.17), we obtain:

$$\dot{\mu}_{ijk} = \sum_{l=1}^{w} i(x_{o_l} - x_g)^{(i-1)}(\dot{x}_{o_l} - \dot{x}_g) + j(y_{o_l} - y_g)^{(j-1)}(\dot{y}_{o_l} - \dot{y}_g) + k(z_{o_l} - z_g)^{(k-1)}(\dot{z}_{o_l} - \dot{z}_g)$$

which can be written using (7.1), (7.2) and (7.3) as follows:

$$\dot{\mu}_{ijk} = \boldsymbol{L}_{\mu_{ijk}}\dot{\boldsymbol{x}}_m + \boldsymbol{b}_{\mu_{ijk}} \tag{7.4}$$

with

$$\boldsymbol{L}_{\mu_{ijk}} = \Big[\underbrace{\boldsymbol{L}_{\mu_{ijk}}(1,1:4)}_{1\text{x}3} \quad \ldots \quad \boldsymbol{L}_{\mu_{ijk}}(1,3p+1:3p+4) \quad \ldots \quad \boldsymbol{L}_{\mu_{ijk}}(1,3M-2:3M+1)\Big] \tag{7.5}$$

$$\begin{aligned}\boldsymbol{b}_{\mu_{ijk}} = \sum_{l=1}^{w} \Bigg[& i(x_{o_l}-x_g)^{(i-1)} \left(\boldsymbol{b}_{o_l}(1) - \frac{1}{w}\sum_{q=1}^{w}\boldsymbol{b}_{o_q}(1)\right) \\ & + j(y_{o_l}-y_g)^{(j-1)} \left(\boldsymbol{b}_{o_l}(2) - \frac{1}{w}\sum_{q=1}^{w}\boldsymbol{b}_{o_q}(2)\right) + k(z_{o_l}-z_g)^{(k-1)} \left(\boldsymbol{b}_{o_l}(3) - \frac{1}{w}\sum_{q=1}^{w}\boldsymbol{b}_{o_q}(3)\right)\Bigg]\end{aligned} \tag{7.6}$$

and

$$\begin{aligned}\boldsymbol{L}_{\mu_{ijk}}(1,3p+1:3p+4) = \sum_{l=1}^{w} \Bigg[& i(x_{o_l}-x_g)^{(i-1)} \left(\boldsymbol{A}_{o_l m_{p+1}}(1,:) - \frac{1}{w}\sum_{q=1}^{w}\boldsymbol{A}_{o_q m_{p+1}}(1,:)\right) \\ & + j(y_{o_l}-y_g)^{(j-1)} \left(\boldsymbol{A}_{o_l m_{p+1}}(2,:) - \frac{1}{w}\sum_{q=1}^{w}\boldsymbol{A}_{o_q m_{p+1}}(2,:)\right) \\ & + k(z_{o_l}-z_g)^{(k-1)} \left(\boldsymbol{A}_{o_l m_{p+1}}(3,:) - \frac{1}{w}\sum_{q=1}^{w}\boldsymbol{A}_{o_q m_{p+1}}(3,:)\right)\Bigg]\end{aligned}$$

Then, $\boldsymbol{L}_{\boldsymbol{\mu}}$ and $\boldsymbol{b}_{\boldsymbol{\mu}}$ are obtained as follows:

$$\begin{aligned}\boldsymbol{L}_{\boldsymbol{\mu}} &= \Big[\boldsymbol{L}_{\mu_{100}}\boldsymbol{L}_{\mu_{010}}\boldsymbol{L}_{\mu_{001}}\boldsymbol{L}_{\mu_{110}}\boldsymbol{L}_{\mu_{101}}\boldsymbol{L}_{\mu_{011}}\boldsymbol{L}_{\mu_{200}}\boldsymbol{L}_{\mu_{020}}\boldsymbol{L}_{\mu_{002}}\boldsymbol{L}_{\mu_{300}}\boldsymbol{L}_{\mu_{030}}\boldsymbol{L}_{\mu_{003}}\boldsymbol{L}_{\mu_{140}}\boldsymbol{L}_{\mu_{014}}\boldsymbol{L}_{\mu_{401}}\Big]^T, \\ \boldsymbol{b}_{\boldsymbol{\mu}} &= \Big[\boldsymbol{b}_{\mu_{100}}\boldsymbol{b}_{\mu_{010}}\boldsymbol{b}_{\mu_{001}}\boldsymbol{b}_{\mu_{110}}\boldsymbol{b}_{\mu_{101}}\boldsymbol{b}_{\mu_{011}}\boldsymbol{b}_{\mu_{200}}\boldsymbol{b}_{\mu_{020}}\boldsymbol{b}_{\mu_{002}}\boldsymbol{b}_{\mu_{300}}\boldsymbol{b}_{\mu_{030}}\boldsymbol{b}_{\mu_{003}}\boldsymbol{b}_{\mu_{140}}\boldsymbol{b}_{\mu_{014}}\boldsymbol{b}_{\mu_{401}}\Big],^T\end{aligned}$$

with each element of $\boldsymbol{L}_{\boldsymbol{\mu}}$ and $\boldsymbol{b}_{\boldsymbol{\mu}}$ obtained directly from (7.5) and (7.6).

Returning back to $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}$, its first eight rows that correspond to the first eight elements of $\boldsymbol{s}$ (5.25) can be obtained as follows:

$$\begin{aligned}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(1,:) &= \Big[1/w, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\Big], \\ \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(2,:) &= \Big[0, 1/w, 0,0,0,0,0,0,0,0,0,0,0,0,0,0\Big], \\ \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(3,:) &= \Big[0,0, 1/w, 0,0,0,0,0,0,0,0,0,0,0,0,0\Big], \\ \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(4,:) &= \Big[0,0,0,0,0,0, 1/w^2, 1/w^2, 1/w^2, 0,0,0\Big],\end{aligned}$$

$$
\begin{aligned}
\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(5,:) &= \left[0,0,0,0,0,0,0,0,0,1/w^2,1/w^2,1/w^2\right], \\
\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(6,:) &= \left[g_6 \quad g_6 \quad g_6 \quad 0 \quad 0 \quad 0 \quad -g_6\frac{\cos\left(\frac{\phi}{3}\right)}{\sqrt{x_1}} \quad 0 \quad g_6\frac{2\sqrt{x_1}\sin\left(\frac{\phi}{3}\right)}{3}\right]\boldsymbol{Q}, \\
\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(7,:) &= \left[g_7 \quad g_7 \quad g_7 \quad 0 \quad 0 \quad 0 \quad g_7\frac{\cos\left(\frac{\phi-\pi}{3}\right)}{\sqrt{x_1}} \quad 0 \quad -g_7\frac{2\sqrt{x_1}\sin\left(\frac{\phi-\pi}{3}\right)}{3}\right]\boldsymbol{Q}, \\
\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(8,:) &= \left[g_8 \quad g_8 \quad g_8 \quad 0 \quad 0 \quad 0 \quad -g_8\frac{\cos\left(\frac{\phi+\pi}{3}\right)}{\sqrt{x_1}} \quad 0 \quad g_8\frac{2\sqrt{x_1}\sin\left(\frac{\phi+\pi}{3}\right)}{3}\right]\boldsymbol{Q}
\end{aligned}
$$

with

$$
\begin{aligned}
g_6 &= \frac{\sqrt{a+b+c-2\sqrt{x_1}\cos(\frac{\phi}{3})}}{2\sqrt{3w}(a+b+c-2\sqrt{x_1}\cos(\frac{\phi}{3}))} \\
g_7 &= \frac{\sqrt{a+b+c+2\sqrt{x_1}\cos(\frac{\phi-\pi}{3})}}{2\sqrt{3w}(a+b+c+2\sqrt{x_1}\cos(\frac{\phi-\pi}{3}))} \\
g_8 &= \frac{\sqrt{a+b+c-2\sqrt{x_1}\cos(\frac{\phi+\pi}{3})}}{2\sqrt{3w}(a+b+c-2\sqrt{x_1}\cos(\frac{\phi+\pi}{3}))}
\end{aligned}
$$

and $\boldsymbol{Q}$ is a 9x15 matrix, whose non-zero elements are given below:

$$
\begin{aligned}
&\boldsymbol{Q}(1,8) = 1\ ,\boldsymbol{Q}(1,9) = 1\ ,\boldsymbol{Q}(2,7) = 1\ ,\boldsymbol{Q}(2,9) = 1\ ,\boldsymbol{Q}(3,7) = 1\ ,\boldsymbol{Q}(3,8) = 1, \\
&\boldsymbol{Q}(4,4) = 1\ ,\boldsymbol{Q}(5,6) = -1,\ \boldsymbol{Q}(6,5) = -1\ ,\boldsymbol{Q}(7,4) = 6\mu_{110}\ ,\boldsymbol{Q}(7,5) = 6\mu_{101}, \\
&\boldsymbol{Q}(7,6) = 6\mu_{011},\ \boldsymbol{Q}(7,7) = -\mu_{002} - \mu_{020} + 2\mu_{200},\ \boldsymbol{Q}(7,8) = -\mu_{002} + 2\mu_{020} - \mu_{200} \\
&\boldsymbol{Q}(7,9) = 2\mu_{002} - \mu_{020} - \mu_{200},\ \boldsymbol{Q}(8,4) = -54\mu_{011}\mu_{101} + 18\mu_{110}(-2\mu_{002} + \mu_{020} + \mu_{200}), \\
&\boldsymbol{Q}(8,5) = -54\mu_{011}\mu_{110} + 18\mu_{101}(\mu_{002} - 2\mu_{020} + \mu_{200}), \\
&\boldsymbol{Q}(8,6) = 18\mu_{011}(\mu_{002} + \mu_{020} - 2\mu_{200}) - 54\mu_{101}\mu_{110}, \\
&\boldsymbol{Q}(8,7) = -18\mu_{011}^2 + 9\mu_{101}^2 + 9\mu_{110}^2 + (-2\mu_{002} + \mu_{020} + \mu_{200})(-\mu_{002} - \mu_{020} + 2\mu_{200}) \\
&\qquad + 2(-2\mu_{002} + \mu_{020} + \mu_{200})(\mu_{002} - 2\mu_{020} + \mu_{200}) \\
&\qquad + (\mu_{002} - \mu_{020} + 2\mu_{200})(\mu_{002} - 2\mu_{020} + \mu_{200}), \\
&\boldsymbol{Q}(8,8) = 9\mu_{011}^2 - 18\mu_{101}^2 + 9\mu_{110}^2 - 2(-2\mu_{002} + \mu_{020} + \mu_{200})(-\mu_{002} - \mu_{020} + 2\mu_{200}) \\
&\qquad - (-2\mu_{002} + \mu_{020} + \mu_{200})(\mu_{002} - 2\mu_{020} + \mu_{200}) \\
&\qquad + (-\mu_{002} - \mu_{020} + 2\mu_{200})(\mu_{002} - 2\mu_{020} + \mu_{200}), \\
&\boldsymbol{Q}(8,9) = 9\mu_{011}^2 + 9\mu_{101}^2 - 18\mu_{110}^2 + (-2\mu_{002} + \mu_{020} + \mu_{200})(-\mu_{002} - \mu_{020} + 2\mu_{200}) \\
&\qquad - (-2\mu_{002} + \mu_{020} + \mu_{200})(\mu_{002} - 2\mu_{020} + \mu_{200})
\end{aligned}
$$

$$- 2(-\mu_{002} - \mu_{020} + 2\mu_{200})(\mu_{002} - 2\mu_{020} + \mu_{200}),$$

$$\boldsymbol{Q}(9,:) = \frac{3x_2}{2x_1\sqrt{4x_1^3 - x_2^2}}\,\boldsymbol{Q}(7,:) + \left(\frac{x_2^2}{4x_1^3\sqrt{4x_1^3 - x_2^2}} - \frac{\sqrt{4x_1^3 - x_2^2}}{4x_1^3}\right)\,\boldsymbol{Q}(8,:)$$

Concerning the last three terms of $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}$, a more complex calculation is required. Since they depend on the rotation matrix (5.18) that corresponds to the ellipsoid orientation, we begin by presenting the derivatives of the inertia matrix eigenvectors. Subsequently, we will obtain the derivative of the rotation matrix, and finally, we will derive the derivative of its $[\boldsymbol{\theta}\mathbf{U}]$ representation (5.20), denoted as $\left[\dot{\boldsymbol{\theta}}\dot{\mathbf{U}}\right]$, which corresponds to the last three terms of $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}$. The derivative of the inertia matrix eigenvectors is obtained from (5.29) as follows:

$$\dot{\boldsymbol{v}}_1(1) = \begin{bmatrix} \frac{1}{f}\left(v_{11} - v_{12} + \frac{1}{3}\right) \\ \frac{1}{f}\left(v_{11} + 2v_{12} + \frac{1}{3}\right) \\ \frac{1}{f}\left(-2v_{11} - v_{12} - \frac{2}{3}\right) \\ \frac{1}{f}\left(-3v_{12}\frac{e}{f} - v_{14}\right) \\ \frac{1}{f}\left(3v_{11}\frac{f}{d} - 3v_{12}\frac{d}{f} - \frac{v_{12}}{v_{11}}\frac{d}{f}\right) \\ \frac{1}{f^2}\left[3v_{11}\frac{ef}{d} - v_{12}v_{13} - \frac{1}{3}v_{15}\right] \\ \frac{\cos\left(\frac{\phi}{3}\right)}{f\sqrt{x_1}}\left(-v_{11} + v_{12} - \frac{1}{3}\right) \\ 0 \\ \frac{2}{3f}\sin\left(\frac{\phi}{3}\right)\sqrt{x_1}\left(v_{11} - v_{12} + \frac{1}{3}\right) \end{bmatrix}\boldsymbol{Q}, \quad \dot{\boldsymbol{v}}_1(2) = \begin{bmatrix} \frac{1}{e}\left(-v_{11} + v_{12}\right) \\ \frac{1}{e}\left(-v_{11} - 2v_{12}\right) \\ \frac{1}{e}\left(2v_{11} + v_{12}\right) \\ \frac{3}{f}v_{12} + \frac{1}{e}v_{14} \\ -\frac{3f}{de}v_{11} + \frac{3d}{ef}v_{12} \\ -\frac{3}{d}v_{11} + \frac{v_{12}v_{13}}{ef} \\ -\frac{v_{12}\cos\left(\frac{\phi}{3}\right)}{e\sqrt{x_1}} + \frac{d\cos\left(\frac{\phi}{3}\right)}{3\sqrt{x_1}v_{16}} \\ 0 \\ 2\sqrt{x_1}\sin\left(\frac{\phi}{3}\right)\left(-\frac{d}{9v_{16}} + \frac{fv_{17}}{9v_{16}^2}\right) \end{bmatrix}\boldsymbol{Q}$$

$$\dot{\boldsymbol{v}}_2(1) = \begin{bmatrix} \frac{1}{f}\left(v_{21} - v_{22} + \frac{1}{3}\right) \\ \frac{1}{f}\left(v_{21} + 2v_{22} + \frac{1}{3}\right) \\ \frac{1}{f}\left(-2v_{21} - v_{22} - \frac{2}{3}\right) \\ \frac{1}{f}\left(-3v_{22}\frac{e}{f} - v_{24}\right) \\ \frac{1}{f}\left(3v_{21}\frac{f}{d} - 3v_{22}\frac{d}{f} - \frac{v_{22}}{v_{21}}\frac{d}{f}\right) \\ \frac{1}{f^2}\left[3v_{21}\frac{ef}{d} - v_{22}v_{23} - \frac{1}{3}v_{25}\right] \\ \frac{\cos\left(\frac{\pi-\phi}{3}\right)}{f\sqrt{x_1}}\left(v_{21} - v_{22} + \frac{1}{3}\right) \\ 0 \\ \frac{2}{3f}\sin\left(\frac{\pi-\phi}{3}\right)\sqrt{x_1}\left(v_{21} - v_{22} + \frac{1}{3}\right) \end{bmatrix}\boldsymbol{Q}, \quad \dot{\boldsymbol{v}}_2(2) = \begin{bmatrix} \frac{1}{e}\left(-v_{21} + v_{22}\right) \\ \frac{1}{e}\left(-v_{21} - 2v_{22}\right) \\ \frac{1}{e}\left(2v_{21} + v_{22}\right) \\ \frac{3}{f}v_{22} + \frac{1}{e}v_{24} \\ -\frac{3f}{de}v_{21} + \frac{3d}{ef}v_{22} \\ -\frac{3}{d}v_{21} + \frac{v_{22}v_{23}}{ef} \\ \frac{v_{22}\cos\left(\frac{\pi-\phi}{3}\right)}{e\sqrt{x_1}} - \frac{d\cos\left(\frac{\pi-\phi}{3}\right)}{3\sqrt{x_1}v_{26}} \\ 0 \\ 2\sqrt{x_1}\sin\left(\frac{\pi-\phi}{3}\right)\left(-\frac{d}{9v_{26}} + \frac{fv_{27}}{9v_{26}^2}\right) \end{bmatrix}\boldsymbol{Q}$$

$$\boldsymbol{v}_3(1) = \begin{bmatrix} \frac{1}{f}\left(v_{31} - v_{32} + \frac{1}{3}\right) \\ \frac{1}{f}\left(v_{31} + 2v_{32} + \frac{1}{3}\right) \\ \frac{1}{f}\left(-2v_{31} - v_{32} - \frac{2}{3}\right) \\ \frac{1}{f}\left(-3v_{32}\frac{e}{f} - v_{34}\right) \\ \frac{1}{f}\left(3v_{31}\frac{f}{d} - 3v_{32}\frac{d}{f} - \frac{v_{32}}{v_{31}}\frac{d}{f}\right) \\ \frac{1}{f^2}\left[3v_{31}\frac{ef}{d} - v_{32}v_{33} - \frac{1}{3}v_{35}\right] \\ \frac{\cos\left(\frac{\pi+\phi}{3}\right)}{f\sqrt{x_1}}\left(-v_{31} + v_{32} - \frac{1}{3}\right) \\ 0 \\ \frac{2}{3f}\sin\left(\frac{\pi+\phi}{3}\right)\sqrt{x_1}\left(v_{31} - v_{32} + \frac{1}{3}\right) \end{bmatrix}\boldsymbol{Q}, \quad \boldsymbol{v}_3(2) = \begin{bmatrix} \frac{1}{e}\left(-v_{31} + v_{32}\right) \\ \frac{1}{e}\left(-v_{31} - 2v_{32}\right) \\ \frac{1}{e}\left(2v_{31} + v_{32}\right) \\ \frac{3}{f}v_{32} + \frac{1}{e}v_{34} \\ -\frac{3f}{de}v_{31} + \frac{3d}{ef}v_{32} \\ -\frac{3}{d}v_{31} + \frac{v_{32}v_{33}}{ef} \\ -\frac{v_{32}\cos\left(\frac{\pi+\phi}{3}\right)}{e\sqrt{x_1}} + \frac{d\cos\left(\frac{\pi+\phi}{3}\right)}{3\sqrt{x_1}v_{36}} \\ 0 \\ 2\sqrt{x_1}\sin\left(\frac{\pi+\phi}{3}\right)\left(-\frac{d}{9v_{36}} + \frac{fv_{37}}{9v_{36}^2}\right) \end{bmatrix}\boldsymbol{Q}$$

$$\boldsymbol{v}_1(3) = \boldsymbol{v}_2(3) = \boldsymbol{v}_3(3) = \boldsymbol{0}$$

with

$$\begin{aligned}
v_{11} &= \frac{de}{-3de + f\left(-a + 2b - c + 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right)} \\
v_{12} &= \frac{ef\left(d\left(-a - b + 2c + 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right) - 3ef\right)}{\left(-3de + f\left(-a + 2b - c + 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right)\right)^2} \\
v_{13} &= a - 2b + c - 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right), \quad v_{16} = \frac{de}{3v_{11}} \\
v_{14} &= \frac{e\left(-a - b + 2c + 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right)}{-3de + f\left(-a + 2b - c + 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right)} \\
v_{15} &= \frac{a}{3} + \frac{b}{3} - 2\frac{c}{3} - \frac{v_{12}}{v_{11}}\frac{ed}{f} - 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right) \\
v_{17} &= -ef + \frac{d}{3}\left(-a - b + 2c + 2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right) \\
v_{21} &= \frac{de}{-3de + f\left(-a + 2b - c - 2\sqrt{x_1}\cos\left(\frac{\pi-\phi}{3}\right)\right)} \\
v_{22} &= \frac{ef\left(d\left(-a - b + 2c - 2\sqrt{x_1}\cos\left(\frac{\pi-\phi}{3}\right)\right) - 3ef\right)}{\left(-3de + f\left(-a + 2b - c - 2\sqrt{x_1}\cos\left(\frac{\pi-\phi}{3}\right)\right)\right)^2} \\
v_{23} &= a - 2b + c + 2\sqrt{x_1}\cos\left(\frac{\pi - \phi}{3}\right), \quad v_{26} = \frac{de}{3v_{21}} \\
v_{24} &= \frac{e\left(-a - b + 2c - 2\sqrt{x_1}\cos\left(\frac{\pi-\phi}{3}\right)\right)}{-3de + f\left(-a + 2b - c - 2\sqrt{x_1}\cos\left(\frac{\pi-\phi}{3}\right)\right)}
\end{aligned}$$

$$
\begin{aligned}
v_{25} &= \frac{a}{3} + \frac{b}{3} - 2\frac{c}{3} - \frac{v_{22}}{v_{21}}\frac{ed}{f} + 2\sqrt{x_1}\cos\left(\frac{\pi-\phi}{3}\right) \\
v_{27} &= -ef + \frac{d}{3}\left(-a-b+2c-2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right) \\
v_{31} &= \frac{de}{-3de + f\left(-a+2b-c+2\sqrt{x_1}\cos\left(\frac{\pi+\phi}{3}\right)\right)} \\
v_{32} &= \frac{ef\left(d\left(-a-b+2c+2\sqrt{x_1}\cos\left(\frac{\pi+\phi}{3}\right)\right)-3ef\right)}{\left(-3de + f\left(-a+2b-c+2\sqrt{x_1}\cos\left(\frac{\pi+\phi}{3}\right)\right)\right)^2} \\
v_{33} &= a-2b+c-2\sqrt{x_1}\cos\left(\frac{\pi+\phi}{3}\right), \quad v_{36} = \frac{de}{3v_{31}} \\
v_{14} &= \frac{e\left(-a-b+2c+2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right)}{-3de + f\left(-a+2b-c+2\sqrt{x_1}\cos\left(\frac{\phi}{3}\right)\right)} \\
v_{35} &= \frac{a}{3} + \frac{b}{3} - 2\frac{c}{3} - \frac{v_{32}}{v_{31}}\frac{ed}{f} - 2\sqrt{x_1}\cos\left(\frac{\pi+\phi}{3}\right) \\
v_{17} &= -ef + \frac{d}{3}\left(-a-b+2c+2\sqrt{x_1}\cos\left(\frac{\pi+\phi}{3}\right)\right)
\end{aligned}
$$

Next, from (5.18) we obtain the derivative of the rotation matrix as:

$$
\dot{\tilde{\boldsymbol{R}}} = \begin{pmatrix}
\frac{\dot{\boldsymbol{v}}_1(1)}{\|\boldsymbol{v}_1\|} - \frac{\boldsymbol{v}_1(1)\|\dot{\boldsymbol{v}_1}\|}{\|\boldsymbol{v}_1\|^2} & \frac{\dot{\boldsymbol{v}}_2(1)}{\|\boldsymbol{v}_2\|} - \frac{\boldsymbol{v}_2(1)\|\dot{\boldsymbol{v}_2}\|}{\|\boldsymbol{v}_2\|^2} & \frac{\dot{\boldsymbol{v}}_3(1)}{\|\boldsymbol{v}_3\|} - \frac{\boldsymbol{v}_3(1)\|\dot{\boldsymbol{v}_3}\|}{\|\boldsymbol{v}_3\|^2} \\
\frac{\dot{\boldsymbol{v}}_1(2)}{\|\boldsymbol{v}_1\|} - \frac{\boldsymbol{v}_1(2)\|\dot{\boldsymbol{v}_1}\|}{\|\boldsymbol{v}_1\|^2} & \frac{\dot{\boldsymbol{v}}_2(2)}{\|\boldsymbol{v}_2\|} - \frac{\boldsymbol{v}_2(2)\|\dot{\boldsymbol{v}_2}\|}{\|\boldsymbol{v}_2\|^2} & \frac{\dot{\boldsymbol{v}}_3(2)}{\|\boldsymbol{v}_3\|} - \frac{\boldsymbol{v}_3(2)\|\dot{\boldsymbol{v}_3}\|}{\|\boldsymbol{v}_3\|^2} \\
-\frac{\|\dot{\boldsymbol{v}_1}\|}{\|\boldsymbol{v}_1\|^2} & -\frac{\|\dot{\boldsymbol{v}_2}\|}{\|\boldsymbol{v}_2\|^2} & -\frac{\|\dot{\boldsymbol{v}_3}\|}{\|\boldsymbol{v}_3\|^2}
\end{pmatrix} \boldsymbol{R}^{*T} \tag{7.7}
$$

with

$$
\begin{aligned}
\|\dot{\boldsymbol{v}_1}\| &= \frac{\boldsymbol{v}_1(1)\dot{\boldsymbol{v}}_1(1) + \boldsymbol{v}_1(2)\dot{\boldsymbol{v}}_1(2)}{\|\boldsymbol{v}_1\|}, \quad \|\dot{\boldsymbol{v}_2}\| = \frac{\boldsymbol{v}_2(1)\dot{\boldsymbol{v}}_2(1) + \boldsymbol{v}_2(2)\dot{\boldsymbol{v}}_2(2)}{\|\boldsymbol{v}_2\|} \\
\|\dot{\boldsymbol{v}_3}\| &= \frac{\boldsymbol{v}_3(1)\dot{\boldsymbol{v}}_3(1) + \boldsymbol{v}_3(2)\dot{\boldsymbol{v}}_3(2)}{\|\boldsymbol{v}_3\|}
\end{aligned}
$$

We can simplify this expression as follows:

$$
\dot{\tilde{\boldsymbol{R}}} = \begin{pmatrix}
\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,1)\boldsymbol{Q} & \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,2)\boldsymbol{Q} & \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,3)\boldsymbol{Q} \\
\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,1)\boldsymbol{Q} & \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,2)\boldsymbol{Q} & \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,3)\boldsymbol{Q} \\
\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,1)\boldsymbol{Q} & \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,2)\boldsymbol{Q} & \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,3)\boldsymbol{Q}
\end{pmatrix} \tag{7.8}
$$

with $\boldsymbol{g}_{\tilde{\boldsymbol{R}}}$ obtained by identifying (7.8) with (7.7). Furthermore, calculating the derivative

of $\left[\boldsymbol{\theta}\dot{\mathbf{U}}\right]$ requires $\dot{\theta}$, which is obtained from (5.21) as:

$$\dot{\theta} = -\left[\left(\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,1) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,2) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,3)\right)/\sin(\theta)\right]\boldsymbol{Q}$$

Finally, we get from (5.22):

$$\left[\boldsymbol{\theta}\dot{\mathbf{U}}\right]_x = \frac{\left(\dot{\tilde{\boldsymbol{R}}}(3,2) - \dot{\tilde{\boldsymbol{R}}}(2,3)\right)2\sin(\theta) - 2\dot{\theta}\cos(\theta)\left(\tilde{\boldsymbol{R}}(3,2) - \tilde{\boldsymbol{R}}(2,3)\right)}{4sin^2(\theta)}$$

from which we obtain:

$$\begin{aligned}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(9,:) = &\frac{\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,2) - \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,3)}{2\sin(\theta)}\boldsymbol{Q} \\ &+ \frac{\cos(\theta)\left(\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,1) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,2) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,3)\right)\left(\tilde{\boldsymbol{R}}(3,2) - \tilde{\boldsymbol{R}}(2,3)\right)}{2\sin^3(\theta)}\boldsymbol{Q}\end{aligned}$$

Similarly, by calculating $\left[\boldsymbol{\theta}\dot{\mathbf{U}}\right]_y$ and $\left[\boldsymbol{\theta}\dot{\mathbf{U}}\right]_z$, we obtain:

$$\begin{aligned}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(10,:) = &\frac{\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,3) - \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,1)}{2\sin(\theta)}\boldsymbol{Q} \\ &+ \frac{\cos(\theta)\left(\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,1) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,2) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,3)\right)\left(\tilde{\boldsymbol{R}}(1,3) - \tilde{\boldsymbol{R}}(3,1)\right)}{2\sin^3(\theta)}\boldsymbol{Q} \\ \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}(11,:) = &\frac{\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,1) - \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,2)}{2\sin(\theta)}\boldsymbol{Q} \\ &+ \frac{\cos(\theta)\left(\boldsymbol{g}_{\tilde{\boldsymbol{R}}}(1,1) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(2,2) + \boldsymbol{g}_{\tilde{\boldsymbol{R}}}(3,3)\right)\left(\tilde{\boldsymbol{R}}(2,1) - \tilde{\boldsymbol{R}}(1,2)\right)}{2\sin^3(\theta)}\boldsymbol{Q}\end{aligned}$$

In conclusion, in this appendix we developed analytically all the rows of the matrix $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}$, which can be concatenated to form the entire matrix $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}$. Furthermore, earlier in this appendix, we analytically derived $\boldsymbol{L}_{\boldsymbol{\mu}}$ (7.5) and $\boldsymbol{b}_{\boldsymbol{\mu}}$ (7.6). Therefore, the interaction matrix $\boldsymbol{L}_{\boldsymbol{s}}$ that relates the variation of the low-dimensional feature vector $\boldsymbol{s}$, given in (5.25), as a function of the manipulated points $\dot{\boldsymbol{x}}_m$, can be analytically calculated using: $\boldsymbol{L}_{\boldsymbol{s}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}\boldsymbol{L}_{\boldsymbol{\mu}}$. Moreover, the term representing the model inertia, $\boldsymbol{b}_{\boldsymbol{s}}$, is also analytically obtained as: $\boldsymbol{b}_{\boldsymbol{s}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\mu}}\boldsymbol{b}_{\boldsymbol{\mu}}$, which concludes our development.

BIBLIOGRAPHY

Aranda, M., Ramon, J. A. C., Mezouar, Y., Bartoli, A., & Özgür, E., (2020), Monocular visual shape tracking and servoing for isometrically deforming objects, *in 2020 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Arriola-Rios, V. E., & Wyatt, J. L., (2017), A multimodal model of object deformation under robotic pushing, *IEEE Transactions on Cognitive and Developmental Systems*, *9*(2), 153–169.

Barber, C. B., Dobkin, D. P., & Huhdanpaa, H., (1996), The quickhull algorithm for convex hulls, *ACM Transactions on Mathematical Software*, *22*(4), 469–483.

Basdogan, C., (2001), Real-time simulation of dynamically deformable finite element models using modal analysis and spectral lanczos decomposition methods. *in Medicine meets virtual reality 2001* (pp. 46–52), IOS Press.

Bender, J., Müller, M., & Macklin, M., (2017), A survey on position based dynamics, 2017, *Proceedings of the European Association for Computer Graphics: Tutorials*, 1–31.

Bhasin, Y., & Liu, A., (2006), Bounds for damping that guarantee stability in mass-spring systems, *Studies in health technology and informatics*, *119*, 55–60.

Bianchi, G., Solenthaler, B., Székely, G., & Harders, M., (2004), Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations, *in Medical image computing and computer-assisted intervention–miccai 2004: 7th international conference, saint-malo, france, september 26-29, 2004. proceedings, part ii 7*, Springer.

Bickel, B., Bächer, M., Otaduy, M. A., Matusik, W., Pfister, H., & Gross, M., (2009), Capture and modeling of non-linear heterogeneous soft tissue, *ACM transactions on graphics (TOG)*, *28*(3), 1–9.

Bourguignon, D., & Cani, M.-P., (2000), Controlling anisotropy in mass-spring systems, *in Computer animation and simulation 2000: proceedings of the eurographics workshop in interlaken, switzerland, august 21–22, 2000*, Springer.

Broyden, C., (2000), On the discovery of the "good broyden" method, *Mathematical programming*, *87*, 209–213.

Caccamo, S., Bekiroglu, Y., Ek, C. H., & Kragic, D., (2016), Active exploration using gaussian random fields and gaussian process implicit surfaces, *in 2016 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Chaumette, F., & Hutchinson, S. A., (2008), Visual servoing and visual tracking, *in Springer handbook of robotics*.

Chi, C., & Berenson, D., (2019), Occlusion-robust deformable object tracking without physics simulation, *in 2019 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Clegg, A., Yu, W., Erickson, Z., Tan, J., Liu, C. K., & Turk, G., (2017), Learning to navigate cloth using haptics, *in Ieee/rsj int. conf. on intelligent robots and systems (iros'17)*.

Coevoet, E., Reynaert, N., Lartigau, E., Schiappacasse, L., Dequidt, J., & Duriez, C., (2015), Registration by interactive inverse simulation: application for adaptive radiotherapy, *International journal of computer assisted radiology and surgery*, *10*, 1193–1200.

Das, J., & Sarkar, N., (2011), Autonomous shape control of a deformable object by multiple manipulators, *Journal of Intelligent & Robotic Systems*, *62*(1), 3–27.

Deledalle, C.-A., Denis, L., Tabti, S., & Tupin, F., (2017), *Closed-form expressions of the eigen decomposition of 2 x 2 and 3 x 3 hermitian matrices* (Doctoral dissertation), Université de Lyon.

de Mathelin, M., & Lozano, R., (1999), Robust adaptive identification of slowly time-varying parameters with bounded disturbances, *Automatica*, *35*(7), 1291–1305.

Dou, M., Davidson, P., Fanello, S. R., Khamis, S., Kowdle, A., Rhemann, C., Tankovich, V., & Izadi, S., (2017), Motion2fusion: real-time volumetric performance capture, *ACM Transactions on Graphics (ToG)*, *36*(6), 1–16.

Duan, Y., Huang, W., Chang, H., Chen, W., Zhou, J., Teo, S. K., Su, Y., Chui, C. K., & Chang, S., (2014), Volume preserved mass–spring model with novel constraints for soft tissue deformation, *IEEE journal of biomedical and health informatics*, *20*(1), 268–280.

Einarsdóttir, H., Guðmundsson, B., & Ómarsson, V., (2022), Automation in the fish industry, *Animal Frontiers*, *12*(2), 32–39.

Etzmuss, O., Gross, J., & Strasser, W., (2003), Deriving a particle system from continuum mechanics for the animation of deformable objects, *IEEE Transactions on Visualization and Computer Graphics*, *9*(4), 538–550.

Ficuciello, F., Migliozzi, A., Coevoet, E., Petit, A., & 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)*, IEEE.

Frank, B., Schmedding, R., Stachniss, C., Teschner, M., & Burgard, W., (2010), Learning the elasticity parameters of deformable objects with a manipulation robot, *in 2010 ieee/rsj international conference on intelligent robots and systems*, IEEE.

Frank, B., Stachniss, C., Schmedding, R., Teschner, M., & Burgard, W., (2014), Learning object deformation models for robot motion planning, *Robotics and Autonomous Systems*, *62*(8), 1153–1174.

Galvez, J. M., & Canton, M., (1993), Normalization and shape recognition of three-dimensional objects by 3d moments, *Pattern Recognition*, *26*(5), 667–681.

Greminger, M. A., & Nelson, B. J., (2008), A deformable object tracking algorithm based on the boundary element method that is robust to occlusions and spurious edges, *International Journal of Computer Vision*, *78*, 29–45.

Güler, P., Pieropan, A., Ishikawa, M., & Kragic, D., (2017), Estimating deformability of objects using meshless shape matching, *in 2017 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Hadfield-Menell, D., Lee, A. X., Finn, C., Tzeng, E., Huang, S., & Abbeel, P., (2015), Beyond lowest-warping cost action selection in trajectory transfer, *in Ieee int. conf. on robotics and automation (icra'15)*.

Han, T., Zhao, X., Sun, P., & Pan, J., (2018), Robust shape estimation for 3d deformable object manipulation, *arXiv preprint arXiv:1809.09802*.

Hauser, K. K., Shen, C., & O'Brien, J. F., (2003), Interactive deformation using modal analysis with constraints.

Hauth, M., Etzmuß, O., & Straßer, W., (2003), Analysis of numerical methods for the simulation of deformable models, *The Visual Computer*, *19*, 581–600.

Higashimori, M., Yoshimoto, K., & Kaneko, M., (2010), Active shaping of an unknown rheological object based on deformation decomposition into elasticity and plasticity, *in 2010 ieee international conference on robotics and automation*, IEEE.

Hosoda, K., & 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)*, https://doi.org/10.1109/IROS.1994.407392

Hu, M.-K., (1962), Visual pattern recognition by moment invariants, *IRE transactions on information theory*, *8*(2), 179–187.

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

Jägersand, M., & Nelson, R., (1996), On-line estimation of visual-motor models using active vision, *image*, *11*, 1, https://api.semanticscholar.org/CorpusID:6886542

James, D. L., & Pai, D. K., (1999), Artdefo: accurate real time deformable objects, *in Proceedings of the 26th annual conference on computer graphics and interactive techniques*.

Kazhdan, M., Bolitho, M., & Hoppe, H., (2006), Poisson surface reconstruction, *in* 4*th eurographics symp. on geometry processing*.

Kinio, S., & Patriciu, A., (2012), A comparative study of h∞ and pid control for indirect deformable object manipulation, *in Ieee int. conf. on robotics and biomimetics (robio)*, IEEE.

Lagneau, R., Krupa, A., & Marchal, M., (2020a), Active deformation through visual servoing of soft objects, *in 2020 ieee international conference on robotics and automation (icra)*, IEEE.

Lagneau, R., Krupa, A., & Marchal, M., (2020b), Automatic shape control of deformable wires based on model-free visual servoing, *IEEE Robotics and Automation Letters*, *5*(4), 5252–5259.

Lehnert, C., English, A., McCool, C., Tow, A. W., & Perez, T., (2017), Autonomous sweet pepper harvesting for protected cropping systems, *IEEE Robotics and Automation Letters*, *2*(2), 872–879, https://doi.org/10.1109/LRA.2017.2655622

Levi, Z., & Gotsman, C., (2014), Smooth rotation enhanced as-rigid-as-possible mesh animation, *IEEE transactions on visualization and computer graphics*, *21*(2), 264–277.

Lloyd, B., Székely, G., & Harders, M., (2007), Identification of spring parameters for deformable object simulation, *IEEE Trans. on Visualization and Computer Graphics*, *13*, 1081–94.

Macklin, M., Müller, M., Chentanez, N., & Kim, T.-Y., (2014), Unified particle physics for real-time applications, *ACM Transactions on Graphics (TOG)*, *33*(4), 1–12.

Makiyeh, F., Chaumette, F., Marchal, M., & Krupa, A., (2023), Shape Servoing of a Soft Object Using Fourier Series and a Physics-based Model, *in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'23*, Detroit, USA.

Makiyeh, F., Marchal, M., Chaumette, F., & Krupa, A., (2022), Indirect positioning of a 3d point on a soft object using rgb-d visual servoing and a mass-spring model, *in 2022 17th international conference on control, automation, robotics and vision (icarcv)*, https://doi.org/10.1109/ICARCV57592.2022.10004285

Massoud Farahmand, A., Shademan, A., & Jagersand, M., (2007), Global visual-motor estimation for uncalibrated visual servoing, *in 2007 ieee/rsj international conference on intelligent robots and systems*, IEEE.

Moré, J. J., (2006), The levenberg-marquardt algorithm: implementation and theory, *in Numerical analysis: proceedings of the biennial conference held at dundee, june 28–july 1, 1977*, Springer.

Morris, D., & Salisbury, K., (2008), Automatic preparation, calibration, and simulation of deformable objects, *Computer methods in biomechanics and biomedical engineering*, *11*(3), 263–279.

Müller, M., & Gross, M. H., (2004), Interactive virtual materials., *in Graphics interface*.

Myronenko, A., & Song, X., (2010), Point set registration: coherent point drift, *IEEE transactions on pattern analysis and machine intelligence*, *32*(12), 2262–2275.

Natsupakpong, S., & Çavuşoğlu, M. C., (2010), Determination of elasticity parameters in lumped element (mass-spring) models of deformable objects, *Graphical Models*, *72*(6), 61–73.

Navarro-Alarcon, D., & Liu, Y.-H., (2017), Fourier-based shape servoing: a new feedback method to actively deform soft objects into desired 2-d image contours, *IEEE Transactions on Robotics*, *34*(1), 272–279.

Navarro-Alarcon, D., Liu, Y.-h., Romero, J. G., & Li, P., (2014), On the visual deformation servoing of compliant objects: uncalibrated control methods and experiments, *The International Journal of Robotics Research*, *33*(11), 1462–1480.

Navarro-Alarcon, D., Liu, Y.-H., Romero, J. G., & Li, P., (2013), Model-free visually servoed deformation control of elastic objects by robot manipulators, *IEEE Transactions on Robotics*, *29*(6), 1457–1468.

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

Nelder, J. A., & Mead, R., (1965), A simplex method for function minimization, *The computer journal*, *7*(4), 308–313.

Nurnberger, A., Radetzky, A., & Kruse, R., (1998), A problem specific recurrent neural network for the description and simulation of dynamic spring models, *in 1998 ieee international joint conference on neural networks proceedings. ieee world congress on computational intelligence (cat. no. 98ch36227)*, IEEE.

Papanikolopoulos, N. P., Khosla, P. K., & Kanade, T., (1993), Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision, *IEEE transactions on robotics and automation*, *9*(1), 14–35.

Pentland, A., & Williams, J., (1989), Good vibrations: modal dynamics for graphics and animation, *in Proceedings of the 16th annual conference on computer graphics and interactive techniques*.

Petit, A., Cotin, S., Lippiello, V., & Siciliano, B., (2018), Capturing deformations of interacting non-rigid objects using rgb-d data, *in 2018 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Petit, A., Ficuciello, F., Fontanelli, G. A., Villani, L., & Siciliano, B., (2017), Using physical modeling and rgb-d registration for contact force sensing on deformable objects, *in Icinco 2017-14th international conference on informatics in control, automation and robotics*, ScitePress; Springer.

Petit, A., Lippiello, V., Fontanelli, G. A., & Siciliano, B., (2017), Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot, *Robotics and Autonomous Systems*, *88*, 187–201.

Petit, A., Lippiello, V., & Siciliano, B., (2015), Real-time tracking of 3d elastic objects with an rgb-d sensor, *in 2015 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Qi, J., Ma, G., Zhu, J., Zhou, P., Lyu, Y., Zhang, H., & Navarro-Alarcon, D., (2021), Contour moments based manipulation of composite rigid-deformable objects with finite time model estimation and shape/position control, *IEEE/ASME Transactions on Mechatronics*, *27*(5), 2985–2996.

Romeo, M., Monteagudo, C., & Sánchez-Quirós, D., (2020), Muscle and fascia simulation with extended position based dynamics, *in Computer graphics forum*, Wiley Online Library.

Royer, L., Krupa, A., Dardenne, G., Le Bras, A., Marchand, E., & Marchal, M., (2017), Real-time target tracking of soft tissues in 3d ultrasound images based on robust visual information and mechanical simulation, *Medical image analysis*, *35*, 582–598.

Rusu, R. B., & Cousins, S., (2011), 3d is here: point cloud library (pcl), *in Ieee international conference on robotics and automation (icra)*.

Schulman, J., Gupta, A., Venkatesan, S., Tayson-Frederick, M., & Abbeel, P., (2013), A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario, *in Ieee/rsj int. conf. on intelligent robots and systems (iros'13)*.

Schulman, J., Lee, A., Ho, J., & Abbeel, P., (2013), Tracking deformable objects with point clouds, *in 2013 ieee international conference on robotics and automation*, IEEE.

Sengupta, A., Krupa, A., & Marchand, E., (2019), Tracking of non-rigid objects using rgb-d camera, *in 2019 ieee international conference on systems, man and cybernetics (smc)*, IEEE.

Sengupta, A., Lagneau, R., Krupa, A., Marchand, E., & Marchal, M., (2020), Simultaneous tracking and elasticity parameter estimation of deformable objects, *in 2020 ieee international conference on robotics and automation (icra)*, IEEE.

Shademan, A., Farahmand, A.-M., & Jägersand, M., (2010), Robust jacobian estimation for uncalibrated visual servoing, *in 2010 ieee international conference on robotics and automation*, IEEE.

Shah, A. J., & Shah, J. A., (2016), Towards manipulation planning for multiple interlinked deformable linear objects, *in Ieee int. conf. on robotics and automation (icra'16)*, https://doi.org/10.1109/ICRA.2016.7487580

Shahamiri, M., & Jagersand, M., (2005), Uncalibrated visual servoing using a biased newton method for on-line singularity detection and avoidance, *in 2005 ieee/rsj international conference on intelligent robots and systems*, IEEE.

Shetab-Bushehri, M., Aranda, M., Mezouar, Y., & Özgür, E., (2022), As-rigid-as-possible shape servoing, *IEEE Robotics and Automation Letters*, *7*(2), 3898–3905.

Shibata, M., & Hirai, S., (2006), Soft object manipulation by simultaneous control of motion and deformation, *in Ieee int. conf. on robotics and automation (icra)*.

Shin, C., Ferguson, P. W., Pedram, S. A., Ma, J., Dutson, E. P., & Rosen, J., (2019), Autonomous tissue manipulation via surgical robot using learning based model predictive control, *in Ieee int. conf. on robotics and automation (icra'19)*.

Sorkine, O., & Alexa, M., (2007), As-rigid-as-possible surface modeling, *in Symposium on geometry processing*, Citeseer.

Taha, A. A., & Hanbury, A., (2015), An efficient algorithm for calculating the exact hausdorff distance, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *37*(11), 2153–2163, https://doi.org/10.1109/TPAMI.2015.2408351

Tang, T., Liu, C., Chen, W., & Tomizuka, M., (2016), Robotic manipulation of deformable objects by tangent space mapping and non-rigid registration, *in Ieee/rsj int. conf. on intelligent robots and systems (iros'16)*.

Tang, T., Wang, C., & Tomizuka, M., (2018), A framework for manipulating deformable linear objects by coherent point drift, *IEEE Robotics and Automation Letters*, *3*(4), 3426–3433.

Tang, Y.-M., Zhou, A., & Hui, K., (2006), Comparison of fem and bem for interactive object simulation, *Computer-Aided Design*, *38*(8), 874–886.

Teschner, M., Heidelberger, B., Muller, M., & Gross, M., (2004), A versatile and robust model for geometrically complex deformable solids, *in Proceedings computer graphics international, 2004.* IEEE.

Tian, Y., Yang, Y., Guo, X., & Prabhakaran, B., (2013), Haptic-enabled interactive rendering of deformable objects based on shape matching, *in 2013 ieee international symposium on haptic audio visual environments and games (have)*, IEEE.

Tonnesen, D. L., (1998), *Dynamically coupled particle systems for geometric modeling, reconstruction, and animation*, University of Toronto.

Trinh, S., Spindler, F., Marchand, E., & Chaumette, F., (2018), A modular framework for model-based visual tracking using edge, texture and depth features, *in 2018 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Wada, T., Hirai, S., Kawamura, S., & Kamiji, N., (2001a), Robust manipulation of deformable objects by a simple pid feedback, *in Ieee int. conf. on robotics and automation (icra)*.

Wada, T., Hirai, S., Kawamura, S., & Kamiji, N., (2001b), Robust manipulation of deformable objects by a simple PID feedback, *in Ieee int. conf. on robotics and automation (icra'01)*, https://doi.org/10.1109/ROBOT.2001.932534

Wada, T., Hirai, S., & Kawamura, S., (1998), Indirect simultaneous positioning operations of extensionally deformable objects, *in Proceedings. 1998 ieee/rsj international conference on intelligent robots and systems. innovations in theory, practice and applications (cat. no. 98ch36190)*, IEEE.

Wang, B., Wu, L., Yin, K., Ascher, U. M., Liu, L., & Huang, H., (2015), Deformation capture and modeling of soft objects., *ACM Trans. Graph.*, *34*(4), 94–1.

Xu, L., Lu, Y., & Liu, Q., (2018), Integrating viscoelastic mass spring dampers into position-based dynamics to simulate soft tissue deformation in real time, *Royal Society open science*, *5*(2), 171587.

Yang, P.-C., Sasaki, K., Suzuki, K., Kase, K., Sugano, S., & Ogata, T., (2016), Repeatable folding task by humanoid robot worker using deep learning, *IEEE Robotics and Automation Letters*, *2*(2), 397–403.

Yu, W., Kapusta, A., Tan, J., Kemp, C. C., Turk, G., & Liu, C. K., (2017), Haptic simulation for robot-assisted dressing, *in Ieee int. conf. on robotics and automation (icra'17)*, https://doi.org/10.1109/ICRA.2017.7989716

Zhang, J., Zhong, Y., Smith, J., & Gu, C., (2019), Neural dynamics-based poisson propagation for deformable modelling, *Neural Computing and Applications*, *31*, 1091–1101.

Zhang, Z., Bieze, T. M., Dequidt, J., Kruszewski, A., & Duriez, C., (2017), Visual servoing control of soft robots based on finite element model, *in 2017 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

Zheng, D., Wang, H., Wang, J., Zhang, X., & Chen, W., (2019), Toward visibility guaranteed visual servoing control of quadrotor uavs, *IEEE/ASME Transactions on Mechatronics*, *24*(3), 1087–1095.

Zhu, B., Gu, L., Zhang, J., Yan, Z., Pan, L., & Zhao, Q., (2008), Simulation of organ deformation using boundary element method and meshless shape matching, *in 2008 30th annual international conference of the ieee engineering in medicine and biology society*, IEEE.

Zhu, J., Navarro, B., Fraisse, P., Crosnier, A., & Cherubini, A., (2018), Dual-arm robotic manipulation of flexible cables, *in 2018 ieee/rsj international conference on intelligent robots and systems (iros)*, IEEE.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Asservissement visuel d'objets déformables à partir du modèle masse-ressort

Mot clés : Asservissement visuel, manipulation d'objets déformables, modèle physique, suivi d'objets déformables.

Résumé : La manipulation d'objets rigides a depuis longtemps alimenté l'automatisation, tandis que l'adaptation aux matériaux souples reste un problème ouvert. Cette étude se concentre sur la commande de la forme d'objets déformables en utilisant le modèle masse-ressort (MSM) pour développer des relations analytiques entre les déplacements des points de l'objet et le mouvement des points manipulés. Nous dérivons une loi de commande basée sur cette relation pour positionner avec précision en temps réel plusieurs points de l'objet dans différentes expériences mettant en œuvre divers objets souples, deux manipulateurs robotiques et une caméra RGB-D. Nous proposons en outre deux méthodes de déformation de la forme des objets souples, en utilisant des vecteurs de caractéristiques de faible dimension, en exploitant les descripteurs de Fourier ou les moments 3D, pour paramétrer la surface de l'objet. Diverses expériences confirment l'efficacité de ces paramétrisations dans la commande de la forme. Nous détaillons également la création du modèle, l'estimation de ses paramètres et le suivi des objets déformables, comblant ainsi l'écart entre les déformations du modèle et celles du monde réel.

Title: Vision-based shape servoing of soft objects using mass-spring model

Keywords: Visual servoing, real-time shape control, compliant manipulation, physics-based model, deformable object tracking.

Abstract: Rigid object manipulation has long driven automation, while adapting to soft materials remains an open problem. This study focuses on deformable object shape control, utilizing the mass-spring model for developing analytical relations between object points displacements and manipulator motions. We derive a control law based on this relation for accurate real-time positioning of multiple object points in different experiments employing various soft objects, two robotic manipulators, and a RGB-D camera. We further provide two methods for soft object shape deformation, using low-dimensional feature vectors, leveraging Fourier descriptors or 3D moments, to parametrize the object surface. Various experiments affirm the efficacy of these parameterizations in shape control. We also elaborate on model creation, its parameter estimation, and deformable object tracking, bridging the gap between the model and real-world deformations.