Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in ICT for Internet and Multimrdia

# Synthetic Dataset for Human Robot Collaborative Transportation of Deformable Objects

Master Candidate

**Arezo Haidari**

**Student ID 2004978**

Supervisor

**Prof. Stefano Ghidoni**

**University of Padova**

Co-supervisor

**Dr. Giorgio Nicola**

**STIIMA-CNR**

Academic Year
2023/2024

# Acknowledgments

To begin with, I would like to express my deepest gratitude to my supervisors, Professor Stefano Ghidoni at the University of Padova and Dr. Giorgio Nicola at STIIMA-CNR Milan, for their invaluable guidance and time throughout my thesis journey.

I am also profoundly thankful to Italy and the University of Padova for granting me this incredible opportunity to pursue my master's degree. As an Afghan girl who was not allowed to study in my own country, this chance has been life-changing.

A special thank you goes to Professor Leonardo Badia, who conducted my interview for university admission. His belief in my potential made this all possible.

Finally, I am immensely thankful to my dear family. Their constant support, unconditional love, and steadfast faith in my capabilities have been the foundation of my achievements.

I am truly grateful to each and every person who has played a role in my journey, and I deeply appreciate their kindness and support.

## Abstract

The manipulation of deformable objects, such as ropes, fabrics, and flexible cables, presents unique challenges due to their high degrees of freedom and complex dynamics. This thesis addresses these challenges by providing a comprehensive review and comparative analysis of existing modeling and control strategies. The primary objectives are to identify and categorize various methods used for modeling deformable objects, including analytical, numerical, and data-driven approaches, and to investigate control strategies, emphasizing both model-based and model-free techniques.

For linear objects, methods such as kinematic modeling, elastic mechanics, and energy-based approaches are discussed. Planar objects, like fabrics and clothes, are examined using numerical methods and mass-spring models, while three-dimensional objects are approached with advanced simulation frameworks.

Simulation tools and frameworks, such as MuJoCo, Gazebo, SOFA, Pybullet, and Nvidia Isaac Sim, are reviewed for their capabilities and limitations in replicating real-world scenarios.

Through this comprehensive review, the thesis highlights the strengths, limitations, and suitable applications of various modeling and control techniques, providing insights into the most effective methods for specific types of deformable objects and scenarios.

# Contents

# List of Figures

# 1 Introduction

The manipulation of deformable objects is a rapidly growing area in robotics and automation, driven by the need for more advanced and adaptable robotic systems. Deformable objects, such as ropes, fabrics, and flexible cables, have unique challenges due to their high degrees of freedom and complex dynamics. Unlike rigid objects, which maintain their shape regardless of manipulation, deformable objects can bend, twist, stretch, and compress, making their behavior highly unpredictable and difficult to model accurately. This unpredictability poses significant challenges for robotic systems designed to manipulate these objects, as traditional rigid-body models and control strategies are inadequate.

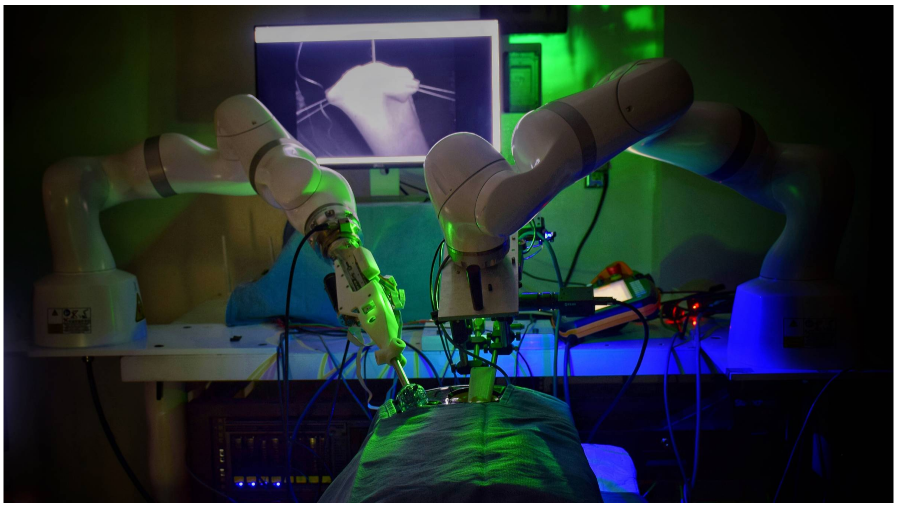


Figure 1.1: A robot laparoscopic surgery on the soft tissue of a pig

The increasing demand for robots capable of handling deformable objects spans various industries and applications. In the medical field, robots that can manipulate surgical sutures or soft tissues are becoming essential. In manufacturing, the ability to handle flexible materials like wires, cables, and textiles is crucial for tasks such as assembly and packaging. Additionally, in the service industry, robots that can fold laundry or handle soft materials are increasingly in demand for domestic and hospitality applications.

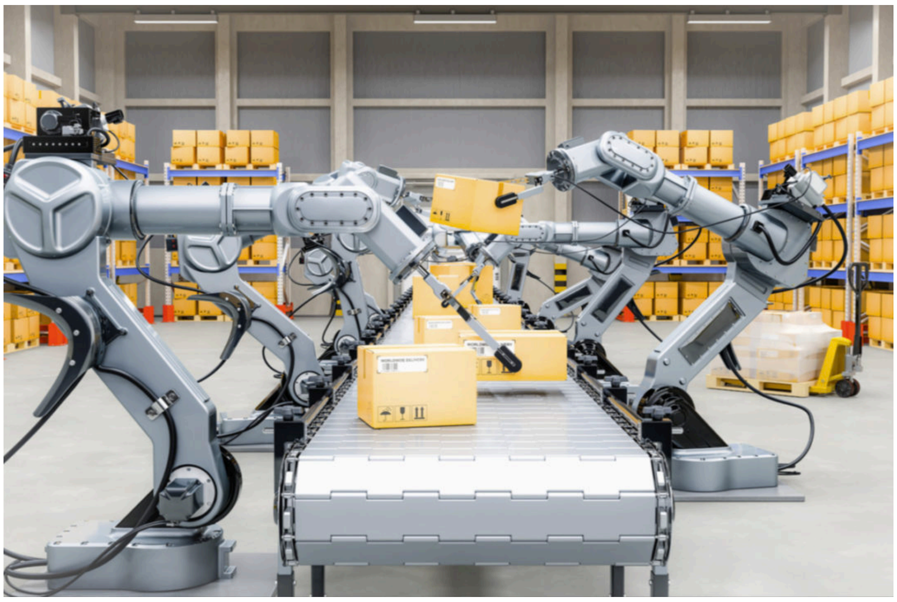


Figure 1.2: Robots increase efficiency and productivity in the warehouse

The applications extend to agriculture, where robots designed to pick fruits and vegetables must handle soft and delicate produce without causing damage. In the textile industry, robots are used for tasks such as cutting, sewing, and ironing, which require precise handling of flexible fabrics.

Manipulating deformable objects needs advanced methods to handle their unpredictable behavior. This thesis delves into these challenges by analyzing a broad spectrum of research papers, existing literature to identify and categorize the various methods used for modeling and controlling of deformable objects. and they categorized based on the geometric complexity of the objects

Figure 1.3: Robots designed to Pick Fruits and Vegetables

## 1.1 Objectives

The primary objectives of this thesis are to provide a comprehensive review of existing literature to identify and categorize the various methods used for modeling deformable objects, which includes analytical, numerical, and data-driven approaches tailored to different shapes and complexities of deformable objects. Additionally, it aims to investigate the control strategies implemented for manipulating deformable objects, emphasizing model-based, model-free approaches. This analysis aims to understand how these strategies are applied to linear, planar, and three-dimensional deformable objects. Furthermore, the thesis aims to provide a comparative evaluation of the different modeling and control techniques, highlighting their strengths, limitations, and suitable applications, and ultimately identify the most effective methods for specific types of deformable objects and scenarios.

## 1.2 Thesis Outline

• Chapter 2: Deformable Object Modeling
This chapter categorizes and details various modeling techniques for deformable objects, including linear, planar, and three-dimensional models.

Each category is further divided into analytical, numerical, and data-driven methods, providing a comprehensive overview of the state-of-the-art modeling approaches.
• Chapter 3: Simulation
This chapter discusses the simulation tools and frameworks used to test and validate the models of deformable objects. Emphasis is placed on the capabilities and limitations of these tools in replicating real-world scenarios.
• Chapter 4: Control Strategies
This chapter focuses on shifts to the control mechanisms designed for manipulating deformable objects. The chapter explores different strategies such as model-based control, model-free control approaches, applied to various shapes of deformable objects.
• Chapter 5: Conclusion
In this chapter the identified challenges and suggested improvements for the manipulation of deformable objects are discussed.

# 2 Deformable Object Modeling

Modeling of deformable objects involves understanding and creating mathematical representations that capture their behavior under external forces and manipulation actions. We categorized the study of modeling based on their shape to linear, planar and three dimensional as different shapes have different levels of geometric complexity and it allows for choosing appropriate modeling techniques that can efficiently capture the geometry of the object it provides the foundation for understanding and predicting how the object will respond to external forces.

## 2.1 Linear

Deformable linear objects (DLOs) are one-dimensional flexible entities like ropes, elastic rods, wires, and cables. There's a growing need to manipulate these objects across various applications, like surgical suturing, leading to considerable research dedicated to robotic solutions for handling them. DLOs in particular are a category of deformable objects with complex dynamics in 3D space due to their twisting and bending behavior. They can move in all three directions in space and deform by bending and twisting due to external forces [28] [26].
This section focuses on the linear category, where the modeling techniques are further subdivided into three main approaches: analytical, numerical, and data-driven methods. Each of these methodologies offers distinct advantages and is

suitable for different types of applications and analysis precision. First we dive deeper into analytical method:

### 2.1.1 Analytical

These methods involve deriving explicit equations that describe the behavior of deformable objects. They often provide exact solutions k, making them highly valuable for theoretical analyses and cases where high precision is required. Analytical methods are typically used when the physical properties and boundary conditions of the system are well-defined and can be accurately modeled by linear or non-linear differential equations.

Modeling the behavior of deformable linear objects (DLOs) such as cables, wires, and ropes is critical for effective control and automation. The complexities inherent in these materials—primarily their flexibility and high degrees of freedom—pose significant challenges for traditional rigid-body modeling techniques.

**Kinematic Model**

To address these challenges, one approach that is assumed is kinematic approach, by focusing on the critical roles of feature points along the DLO.

The kinematic model constructs a mathematical framework that directly correlates the movement of a robot's end effector with the positional and velocity changes of these feature points, using Jacobian matrices to capture this relationship.

In paper [13], the shape of the DLO is represented as a series of feature points, and the positions of feature points are measured with a camera, Fig 2.1.

In general, the velocity of a specific feature point can be related with the velocity of the robot end effector as:

$$x_i = J_i(x_i, r)\dot{r} \tag{2.1}$$

where

$$x_i = [x_{i1}, x_{i2}, \ldots, x_{ik}]^T \in \mathbb{R}^k \tag{2.2}$$

is the position of i-th feature in sensory space (e.g. k=2 for image space,

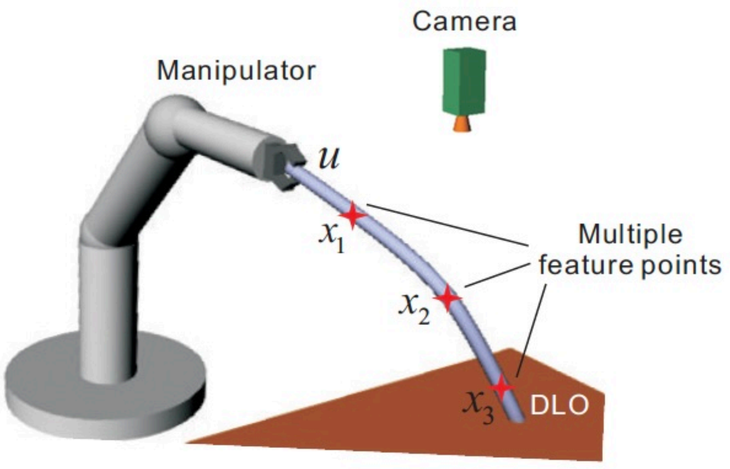


Figure 2.1: DLO representation as a Multiple Feature Points

where i = 1,2...,m, m is the number of multiple features)

$$r = [r_1, r_2, \ldots, r_n]^T \in \mathbb{R}^n \tag{2.3}$$

is the position and orientation of the robot end effector in Cartesian space.

The model allows for real-time control adjustments based on observed changes in feature point positions. While the kinematic model is straightforward and quick to compute that makes it computationally efficient framework for real-time control, it has a significant limitation. It does not take into account the physical properties of the object, such as flexibility, elasticity, stiffness, and weight. It works best when those material characteristics don't really change the way the object needs to be handled. Where the feature points can be accurately tracked. The accuracy of the model is heavily dependent on the visibility and correct identification of feature points along the DLO.

**Elastic Mechanics**

Another approach for modeling of deformable linear object primarily considers flexural deformations. Modeling in the paper is based on the principles of elastic mechanics and the specific assumptions that the object is elastic yet inextensible. Therefore, the object can bend and flex but its length remains constant, this introduced in the paper [9].

The modeling approach focuses on capturing the behavior of the object as it deforms, particularly under the influence of external forces applied by robotic manipulation. The model assumes that all deformations occur in a plane, simplifying the dynamics involved and focusing on the two-dimensional interactions between the object and the robotic manipulator.

A key aspect of the model is the quantification of the object's internal energy as it deforms. The paper specifically discusses the increase in internal energy due to flexural deformation when the object is manipulated. This flexural energy is central to understanding how the object behaves during the flex-and-flip manipulation process.

The model provides a strong foundation for controlling and understanding of elastic, inextensible linear objects in two-dimensional manipulation tasks, but there are materials that might have some amount of stretch under tension and also the limitation to planar deformations ignores out-of-plane forces and deformations. It might not be appropriate to apply to materials with different properties or in scenarios where three-dimensional forces are there.

**Catenary-based Model**

For estimation of deformation state without any contact of strip-like deformable objects in manipulating process, information for tension in the object is required to be able to fit strip-like deformable object in 3D space. And this is done in introduces catenary-based model in the paper [4], this model effectively captures the physical behavior of such objects under their own weight.

A catenary is defined as the curve an idealized hanging chain or cable assumes under its own weight when supported only at its ends. This physical principle is applied to model the deformable strip-like object.

The shape of the catenary is mathematically described by the equation:

$$z = z_{\text{low}} + c\left(\cosh\left(\frac{x - x_{\text{low}}}{c}\right) - 1\right) \tag{2.4}$$

Where: $z$ represents the vertical position of a point on the curve. $x$ is the horizontal position of that point. $z_{\text{low}}$ is the vertical position of the lowest point of the catenary curve. $x_{\text{low}}$ is the horizontal position of the lowest point of the catenary curve. $c$ is the catenary constant, related to the horizontal component of the tension ($T_{H,i}$) at any point $i$ on the catenary, by the relationship $c = T_{H,i}/(\mu g)$, where $\mu$ is the mass per unit length of the object, and $g$ is the acceleration due to gravity.

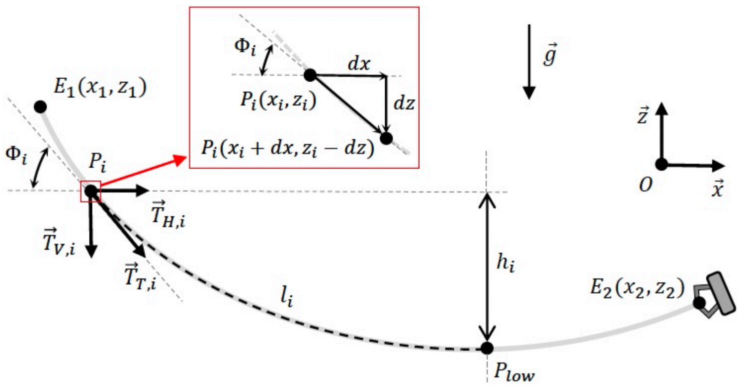


Figure 2.2: Representation of a catenary in the 2D space.

The catenary model is a straightforward way for representing behaviour of strip-like objects under tension, it has real-time estimation that is crucial for applications where immediate feedback and adjustments are necessary. Moreover it is cost effective. But this model is designed for strip-like objects that might not be appropriate for object with other shape and properties. And also the model assumes that the primary mode of deformation is bending under gravity, without significant twisting or compression. Objects that require consideration of other types of deformations (like elastic deformation beyond simple bending, torsional twisting, or volumetric compression) would not be accurately modeled by this model.

**Multiple Interlinked Objects With Mass Points**

Due to the motion characteristics of DLOs, vibration is inevitable at the end of a DLO. Therefore, for reducing the oscillation at the end of DLO, a dynamic model of DLO is represented in the paper [2] as 2nd order underacutated system with n equations and local linearization near equiblirium.
The modeling of the deformable linear object (DLO) is achieved through a detailed process that incorporates the dynamics of interlinked mass points to represent the DLO.

Multiple interlinked objects with mass points, denoted as $P_i$, where each segment of the DLO is modeled with a corresponding mass point. The total length of the DLO is $L$, and the length of each segment is given as $h = \frac{L}{n}$, where $n$ is the number of segments. Each segment is defined by an angle $\theta_i$, which is the angle between the $i$-th segment and the vertical direction.

The dynamic model of the DLO is formulated using the Euler-Lagrange method in a two-dimensional (2D) plane, considering only manipulator motion within this plane. The equation of motion is expressed as:

$$M\ddot{\theta} = \tau - K\theta - \rho h(-CH^T H\tilde{S}\dot{\theta} + S^T HC\tilde{\theta} + \ddot{x}_0 C^T I + \ddot{y}_0 S^T I + ghS^T N) \quad (2.5)$$

where $M = \rho h(CH^T C + S^T HS)$ is the mass matrix. $\theta$ is the vector of segment angles. $\tau$ is the torque from the manipulator. $K$ is the stiffness matrix. $\rho$ is the linear density of the DLO. $C$ and $S$ are matrices composed of the cosine and sine of segment angles, respectively. $\tilde{S}$ and $\tilde{C}$ are the derivative matrices related to $S$ and $C$. $\ddot{x}_0$ and $\ddot{y}_0$ are the accelerations in the $X$ and $Y$ directions. $I$ is a vector of ones, and $N$ is a matrix representing the cumulative length contribution of each segment.

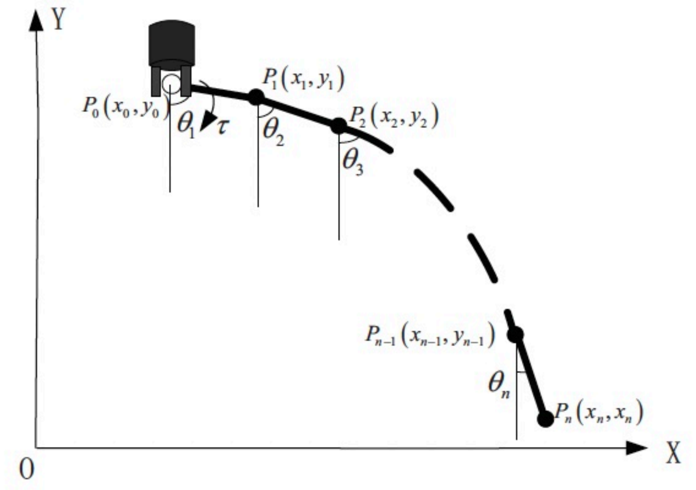


Figure 2.3: The description of a DLO in plane

The paper then discusses the local linearization of the dynamic model near the equilibrium point, assuming small oscillations. This simplifies the model by considering the linear density, the stiffness of the DLO, and gravitational effects, leading to a more manageable form for analysis and control design.

This modeling has detailed representation of the DLO's physical properties along its length. Using the Euler-Lagrange provides capturing the true physical behavior of the DLO. And Linearizing the system near equilibrium simplifies the complex nonlinear dynamics into a more manageable form. It has stiffness matrix K, damping C and S (segment angels) that gives a complete dynamic representation of the DLO. Although it provides detailed modeling, it increases the computational load. The model assumes small oscillations,the downside of this assumption is that if the DLO experiences large movements or swings, the model might not accurately predict or handle these situations.

**DLO Energy Model (Using Mass-Spring Model)**
In the manipulation of Deformable Linear Objects (DLOs) it is crucial to address the challenges presented by their flexibility and complex dynamics. To achieve precise and stable configurations in constrained environments, a DLO energy model introduced in the paper [19].

This model quantifies the potential energy of a DLO based on its configuration. By ensuring that the DLO reaches a stable equilibrium—where it is at a local minimum of energy—the model guarantees stability and minimal structural stress. And also the DLO energy model enables the projection of randomly sampled or initial configurations into stable ones, essential for planning feasible, safe, and stable paths.

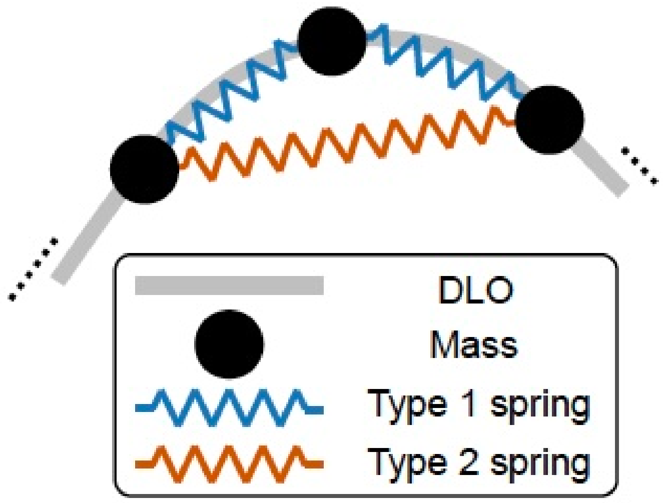


Figure 2.4: Illustration of mass-spring model

This approach uses a mass-spring model to represent the DLO. The mass-

spring model captures the elastic properties and potential energy of the deformable object.
The potential energy of the DLO, indicates as $E$, is a function of the configuration of the object, represented by the positions of the mass points. The paper provides a specific formulation of this energy for a DLO modeled with two types of springs:
Type 1 springs connect adjacent mass points.
Type 2 springs connect every other mass point, adding rigidity and capturing bending energy.

The total energy $E$ of the system is defined as the sum of the energies stored in these springs, formulated as:

$$E = \frac{1}{2}\sum_{k=1}^{m-1} \lambda_1 \left( \|x_{k+1} - x_k\|^2 - \left(\frac{L}{m-1}\right)^2 \right)^2 + \frac{1}{2}\sum_{k=1}^{m-2} \lambda_2 \left( \|x_{k+2} - x_k\|^2 - \left(\frac{2L}{m-1}\right)^2 \right)^2 \tag{2.6}$$

$x_k$ represents the position of the $k^{th}$ mass point. $L$ is the total length of the DLO. $m$ is the number of mass points modeling the DLO. $\lambda_1$ and $\lambda_2$ are the stiffness coefficients for Type 1 and Type 2 springs, respectively. The first sum captures the energy of Type 1 springs and the second sum captures the energy of Type 2 springs.

The stable shapes of the Deformable Linear Object (DLO) are the ones that minimize an energy function while keeping specific mass points (usually the ends of the DLO) fixed. This modeling allows the planning and control of the DLO's shape by manipulating the positions of the mass points to minimize the energy function, which corresponds to moving the DLO towards a desired configuration while taking its elastic properties into account.

The mass-spring model accurately represents its flexible and dynamic nature. It has stability assurance. By ensuring the DLO reaches a stable equilibrium, where the potential energy is at a local minimum, the model make sure the stability. The energy model provides a quantitative way to measure the configuration energy, enabling energy-efficient manipulations. But the calculations required for the energy model, especially in real-time applications, can be computationally intensive due to the need to solve for the minimum energy states repeatedly. Moreover, the basic mass-spring model might not adequately handle very complex deformations or interactions with multiple obstacles,

### 2.1.2 Data-Driven

Another way to model deformable objects is through a data-driven approach, using computational models that learn from data. This method is especially useful when the physics involved is not well understood, or when traditional modeling is impractical due to the object's extreme complexity or variable material properties. Machine learning models, like neural networks, can be trained on experimental or simulation data to predict the behavior of deformable objects. These models can adapt to new data and manage nonlinearities and complex interactions that are challenging for other modeling techniques.

**Interaction Networks (INs)**
Due to the high number of degrees of freedom and complex behaviors of linear deformable objects like bending and twisting, a technique based on Interaction Networks (IN) is introduced in paper [27] to capture the complex dynamics and interactions between different segments of a DLO as it offers a framework for learning the dynamics of systems composed of multiple interacting components, allows to leverage machine learning techniques to approximate the dynamics of DLOs directly from data.

DLO described as a series of connected segments based on an explicitly discretized Cosserat rod. A neural network structure and a function approximator is proposed ${}^f\theta$.
The function approximator can predict the future state, $s_{t+1}$ by giving current state $s_t$ and current action $a_t$. i.e.

$$s_{t+1} = \hat{\theta}^f(s_t, a_t) \tag{2.7}$$

$s_t$ contains position and orientation of the segment in DLO.

The IN method assumes that the overall dynamics of the particle-based system are made up of local interactions between related particles. Therefore, it is a generic dynamics model for particle-based systems represented by a directed graph, G =(V, E).
The graph for a DLO is a chain with vertices representing different segments of

the DLO connected by edges. The vertices, $v \in \mathbb{V}$, represent the particles. The edges $e \in \mathbb{E}$ represent their relations encoded by

$$e_{ij} = (v_i, a_i, v_j, a_j) \tag{2.8}$$

a$_i$ is the external force applied on vertex i. A simple formulation of is directly concatenating vi, ai, vj , and aj. A directed edge, $e_{ij}$, represents a relation where $v_i$ is the receiver and $v_j$ the sender. Therefore the dynamics model $^f\theta$ based on the IN method maps a state:

$$s_t = \left(\{v_{i,t} \mid v_i \in V\}, \{e_{i,j,t} \mid e_{ij} \in E\}\right) \tag{2.9}$$

The structure of the model is represent in Fig 2.5.

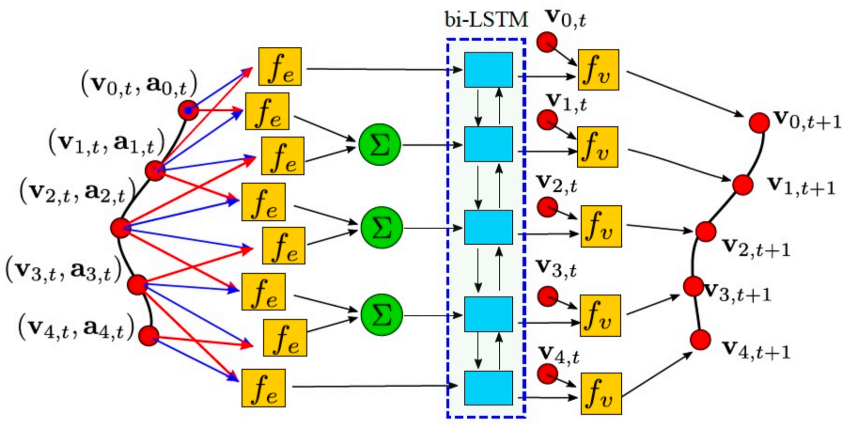


Figure 2.5: Neural network structure for modeling a DLO

The vertices $v_{i,t}$ correspond intersections between neighbor Cosserat rod segments. $a_{i,t}$ represents the action applied on the vertices. $f_e$ is the relation-centric network and $f_v$ is the object-centric network.
By using Interaction Networks, the model can get the complex behaviors and interactions among the segments of a DLO, such as bending and twisting. The model use machine learning techniques to learn the dynamics directly from data, rather than relying only on pre-defined physical models. This can improve the model's accuracy and adaptability to new situations. However it can be computationally expensive especially the neural networks that handling complex interactions like those in INs. Quality and quantity of training data can effect the effectiveness of the model.

**Attention-Based Global Deformation Model**

Another way for modeling of linear deformable object is based on attention mechanism in GDM (Global Deformation Model) learning that is represented in the paper [24], by capturing the spatial relationship between the feature points. The DLOs are represented through a set of $n$ feature points along the object, where the position of each feature point $x_i$ contributes to the overall state representation of the DLO. The state of the DLO is represented by a stacked vector $x \in \mathbb{R}^{3n}$, encapsulating the positions of all feature points in three-dimensional space. This representation facilitates the modeling of the DLO's deformation dynamics.

The attention-based global deformation model (GDM), described by the equation:

$$x = F(q, u) \tag{2.10}$$

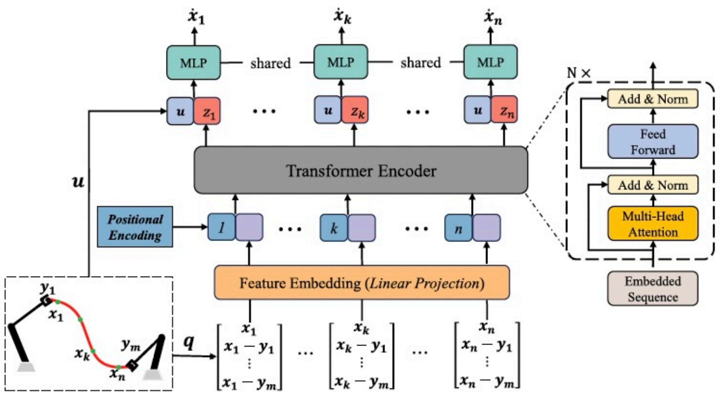


Figure 2.6: Attention-based model architecture for global DLO deformation model learning.

Here, $x$ represents the velocity of the DLO's feature points, $F(\cdot, \cdot)$ is the learned global deformation model, $q$ is the system configuration combining the DLO's state and the manipulators' configurations, and $u$ is the control input, i.e., the velocities of the end-effectors (EEs) manipulating the DLO. This model leverages an attention mechanism to capture the complex spatial relationships among the DLO's feature points, which significantly enhances the model's ability to predict the DLO's behavior accurately.

It has precise modeling by focusing on crucial spatial relationships among feature points, increasing prediction accuracy. However, it requires significant computational resources, relies heavily on extensive, high-quality training data.

### 2.1.3 Numerical

The numerical method in the context of modeling deformable objects involves using computational techniques to approximate the physical laws that control how these objects deform. This approach is particularly useful when analytical solutions are too complex or impossible to derive due to the complicated geometries, nonlinear material properties, or complicated boundary conditions involved.
Numerical methods are essential for handling real-world applications where precision and adaptability to diverse modeling conditions are required. They enable detailed investigations of how deformable objects will perform under various loads and conditions, providing crucial insights that are not readily obtainable through purely analytical methods.

**Location-Based Point Chain (LBPC) Model**
One technique used for modeling deformable linear objects (DLOs) is within the context of reinforcement learning, particularly using PILCO (Probabilistic Inference for Learning Control). This approach focuses on the Location-Based Point Chain (LBPC) model, as detailed in the paper [5]. By assuming a sequence of control points to represent the rope's shape and configuration, this model facilitates a precise understanding and manipulation of DLOs in two-dimensional space.

In this paper, the rope is modeled using the Location-Based Point Chain (LBPC) approach that simplifies the complex geometry of DLOs to manageable data for the model. This means that the configuration of the rope, is described and represented by a sequence of control points in the XY-plane. These control points effectively capture the shape and configuration of the rope.
Each control point corresponds to a specific location along the rope's length. At step t, the DLO configuration is specified by the absolute location of the first

control point $p_{0,t}$ and the relative location between first control point to others as

$$v_{i,t}, \quad i = 1, 2, \ldots, n \tag{2.11}$$

The action is the movement from $p_{0,t}$ to $p_{0,t+1}$. The shape of DLO is approximated by n sequesnce control points $\{p_0, p_1, ..., p_n\}$ in the XY-plane as shown in Fig 2.5.

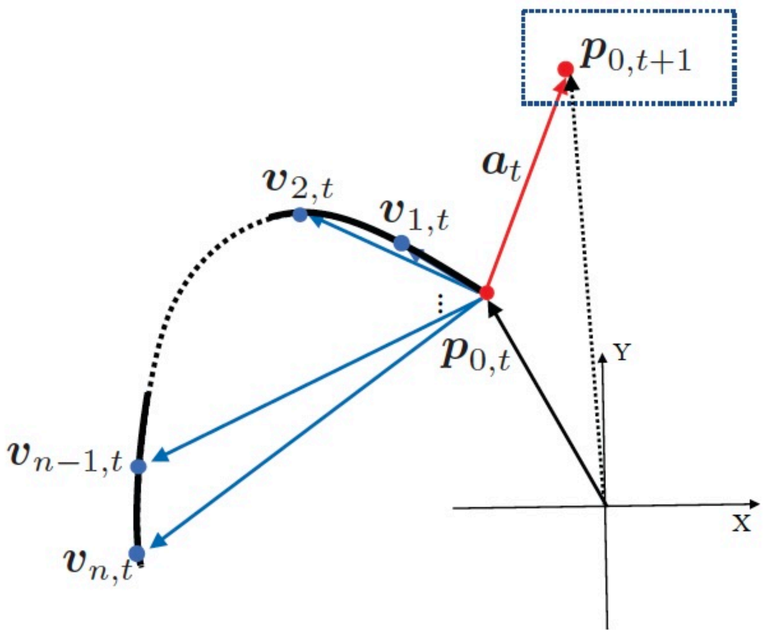


Figure 2.7: Location-Based Point Chain Model for DLO

The Cartesian coordinate for $p_i$ at step t is

$$p_{i,t} = [x_{i,t}, y_{i,t}]^T, \quad i \in \{0, 1, 2, \ldots, n\} \tag{2.12}$$

that $p_0$ is the control point that represents a DLO end. Initially, the DLO is defined as a straight line, such that all control points are evenly distributed, i.e.

$$p_i - p_{i-1} = p_j - p_{j-1}, \quad p_i - p_j = (i-j)(p_i - p_{i-1}), \quad i, j \in \{1, 2, \ldots, n\} \tag{2.13}$$

Thus, the state at step t is defined by control points as:

$$\mathrm{st} = \left[\left(p_0^T, v_1^T, v_2^T, \ldots, v_{n-1}^T, v_{n,t}^T\right)\right]^T \tag{2.14}$$

where

$$v_{i,t} = p_{i,t} - p_{0,t}, i = 1, 2..., n \tag{2.15}$$

The LBPC modeling approach for deformable linear objects (DLOs) has several advantages. It simplifies the complex geometry of DLO to a sequence of control points that make it easier to handle the manipulation. It had flexibility. By using specific points (control points) to represent these objects, the model can easily change and adjust to different shapes and positions. The approach is limited to two-dimensional space. Although this simplifies computations, it may not get the full complexity of DLO manipulation in three-dimensional environments, which are common in practical applications. Moreover, The behavior of DLOs under external forces is often non-linear, like small changes in force can result in large, unpredictable changes in shape. This linear and simplified approach can not manage these non-linearities well.

**Discrete Differential Geometry**

In the research paper [11], the authors propose a novel approach to model the deformation of elastoplastic Deformable Linear Objects (DLOs) by employing numerical methods rooted in discrete differential geometry. This modeling framework is particularly focused on capturing the intrinsic geometrical properties of DLOs, such as curvature and torsion, which are essential for understanding and manipulating the complex behaviors of these materials.
The curvature ($\kappa$) is calculated using the formula:

$$\kappa_i = \frac{2\tan(\theta_i/2)}{l} \approx \frac{\theta_i}{l} \tag{2.16}$$

where $l$ is the length of a segment, and $\theta_i$ is the angle between the tangent vectors of two consecutive segments.
Torsion ($\tau$), which measures the rate of twist along the DLO, is approximated by

$$\tau_i = \frac{2\tan(\phi_i/2)}{l} \approx \frac{\phi_i}{l} \tag{2.17}$$

with $\phi_i$ being the angle between two consecutive binormal vectors.

Proposed approach here is offering precision in modeling, using intrinsic geometrical properties such as curvature and torsion allows for a detailed and precise representation of the DLOs' shape. This can help accurately capturing the bending and twisting behaviors of the objects. It also provides a solid mathematical foundation, therefore it is ensuring the robustness and accuracy of the modeling process. By considering both curvature and torsion it can also covers both bending and twisting behaviors of the DLO. It has challenges related to computational complexity. The detailed calculations of curvature, torsion, and related geometrical properties can be computationally intensive, and make slowing down the real-time applications.

**Compliant Position-Based Dynamics (XPBD)**
In manipulation of deformable objects the ability to accurately model the dynamic behavior of materials such as ropes presents unique challenges. Traditional models often struggle with computational efficiency and real-time interaction, which are crucial for applications ranging from automated suturing to cable manipulation in industrial settings. To address these challenges, in the paper [14] the compliant position-based dynamics (XPBD) framework offers a robust solution. The following description outlines how XPBD is applied to model rope-like objects effectively, providing a detailed look at the specific constraints and methodologies utilized to replicate the physical properties of these objects.

This approach discretizes the object into a sequence of particles connected by constraints that model the physical behavior of the rope, including stretching, bending, and twisting.

The rope-like object is represented as a series of discrete particles with Cartesian coordinates $x \in \mathbb{R}^3$. The orientations between adjacent particles are described using quaternions

$$q = [q_w, q_v] \in SO(3) \tag{2.18}$$

facilitating the handling of bending and twisting deformations. This discretization allows for the modeling of the rope's physical properties using geometric constraints within the XPBD framework.

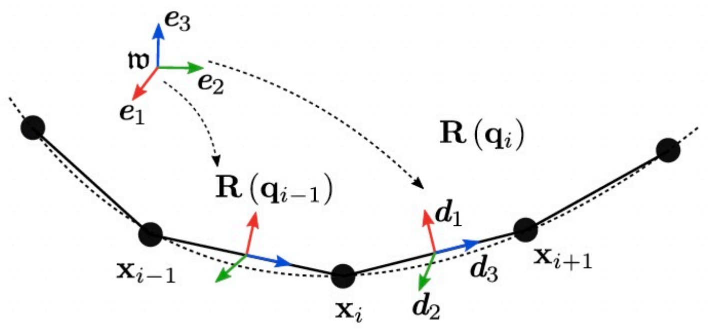


Figure 2.8: Proposed geometric model of rope-like objects

Shear and Stretch Constraint ensure that the stretch or compression of the rope segments relative to their rest state is accounted for, maintaining the inextensibility property. This is expressed as:

$$C_S(x, q) = \{c_{S_i}(x_i, x_{i+1}, q_i) \mid i \in [1, 2, ..., N-1]\} \tag{2.19}$$

where $c_{S_i}$ integrates shear and stretch deformations for each pair of neighboring particles, $x_i$ and $x_{i+1}$, and $q_i$ represents the orientation affecting shear strain.

Bend and Twist Constraint evaluate the rod's bending and twisting by comparing the orientation of adjacent segments against their rest configuration. It's formulated as:

$$C_B(q) = \{c_{B_i}(q_i, q_{i+1}) \mid i \in [1, 2, ..., N-1]\} \tag{2.20}$$

where $c_{B_i}$ calculates the difference in orientation between neighboring segments to represent bending and twisting.

Using geometric constraint gives a a highly accurate representation of the rope's physical properties, including stretching, bending, and twisting. This method is computationally efficient. The iterative solver within XPBD allows for stable and quick convergence, enabling robots to interact with deformable objects in real time without significant delays. But very long or detailed simulations it can increase computational complexity (i.e.high-detail requirements.) And Proper tuning of constraint parameters is important for maintaining accuracy and stability. Then finding the optimal parameters involves many trial and error, that can be time-consuming. Therefore, XPBD strikes a balance between computational efficiency and accuracy.

**Position-Based Dynamics (PBD)**

Position-Based Dynamics (PBD) is utilized as a method for modeling deformable objects in the paper [1], such as 1D ropes and 2D cloths. This approach is highlighted for its efficiency in simulating the flexible dynamics of these objects, which is crucial for real-time applications in robotic manipulation.

The core of the deformable object modeling involves representing these objects as a collection of discrete particles. In the case of 1D objects, like ropes, an object of length $L$ is discretized into $N$ line segments, each of length

$$l_i = \frac{L}{N} \tag{2.21}$$

where $i$ ranges over the set $\{1, 2, \ldots, N\}$.

The state of a deformable object is denoted as $x \in \mathbb{R}^{3N}$, encapsulating the 3D locations of the particles or the center positions of the line segments. This state representation facilitates the correlation between agent interactions and particle positions, thereby directly influencing the simulation of the object's deformable behavior.

The implementation of PBD within the paper incorporates advanced techniques to accurately simulate the physical properties of deformable objects:

Each particle's velocity is updated based on external forces and mass, followed by an update of the particle's position based on the new velocity.

A comprehensive constraint that includes stretch, bending, and twisting is applied to model the object's deformable properties accurately.

Post-constraint solving, velocity is adjusted to reflect the constraints' effects, ensuring the simulation remains consistent with physical behaviors.

Parameters such as Young's modulus, torsion modulus, and zero-stretch stiffness are critical for modeling the object's behavior.

PBD is computationally efficient by simplifying the complex dynamics of deformable objects into discrete particles and also it is suitable for real-time task. While PBD is efficient, it involves approximations that may not capture all the details of a deformable object's behavior. Then high-accurate simulations may require more detailed methods.

**Discretizing the DLO Into A Chain of Connected Nodes**

In the modeling of the deformable objects, by breaking down the continuous form of a DLO into a chain of uniformly spaced nodes, this method simplifies the infinite-dimensional problem into a tractable, finite-dimensional space. And discretization serves as a foundational technique for addressing the complexities of robotic manipulation. It allows for efficient computation while handling the dynamic and flexible nature of DLOs during robotic interactions. This model is chosen to facilitate the application of the Coherent Point Drift (CPD) method in the paper[23].

The DLO is modeled through a process of discretization, transforming the continuous entity into a series of discrete elements that can be more easily managed and manipulated by computational algorithms. This model is essential for enabling the subsequent steps of state estimation, task planning, and trajectory planning within the proposed framework.

The modeling process begins by discretizing the deformable linear object into a chain of connected nodes, spaced at uniform intervals. This approach transforms the DLO, which naturally exists as a continuous entity with an infinite-dimensional configuration space, into a finite set of discrete points. Each node in the chain represents a segment of the DLO and is defined by its position in a three-dimensional Cartesian space.

By representing the DLO as a series of nodes, the model simplifies the complex geometry of the object into a manageable form for computational processing. This discretization is a crucial step that allows for the real-time tracking of the object's state as it undergoes various manipulations.

Each node in the discretized model is represented by its coordinates in a three-dimensional space, effectively capturing the configuration of the DLO at any given moment. The position of each node, denoted as $x_n \in \mathbb{R}^3$, where $n$ is the node index, is subject to change as the object is manipulated. The series of nodes, collectively representing the state of the DLO, is denoted as $X = \{x_1, x_2, \ldots, x_N\}$, where $N$ is the total number of nodes. This representation captures the spatial configuration of the DLO and serves as the basis for the state estimation process, where the goal is to track the position of each node over time accurately.

**Handling Real-Time Dynamics**
The dynamic nature of DLOs, which can change shape in complex ways due to their high degrees of freedom, poses significant challenges for real-time tracking and manipulation. The discretized model of the DLO, with its series of nodes, enables the application of the Coherent Point Drift (CPD) method for state estimation. CPD is a non-rigid point set registration technique that aligns two sets of points (the model and the observed point cloud data) in a way that accounts for the deformable nature of the object. By applying CPD, the framework can robustly estimate the real-time state of the DLO, even in the presence of noise, outliers, and occlusions in the observed point cloud data.

**Energy Based Model**
A model represents the DLO as a series of interconnected rigid segments, known as kinematic chain model that each segment is joined by ball joints that allow for rotational degrees of freedom, then approximating the flexible nature of the object. Inside each ball joint, friction and damping forces are modeled to more accurately reflect the physical behavior of real DLOs. This includes considerations for how the segments of the DLO resist motion relative to each other.
This kinematic chain model introduced in the paper [29].

The position and orientation of any target point along the DLO are determined by the transformation from the world coordinate system to that point, represented as $T_0 \rightarrow T_P(q)$. This transformation is calculated using forward kinematics, which involves multiplying the transformations between each joint's coordinate system (COS):

$$T_0 \rightarrow T_P(q) = \prod_{i=0}^{v-1} T_i \rightarrow i + 1(q) \cdot T_v \rightarrow T_P(q) \tag{2.22}$$

where $v$ is the joint number of the target point, and $T_v \rightarrow T_P(q)$ is an arbitrary affine transformation from the joint to the point of interest on the body.

The model's accuracy in representing the DLO's curvature is determined by

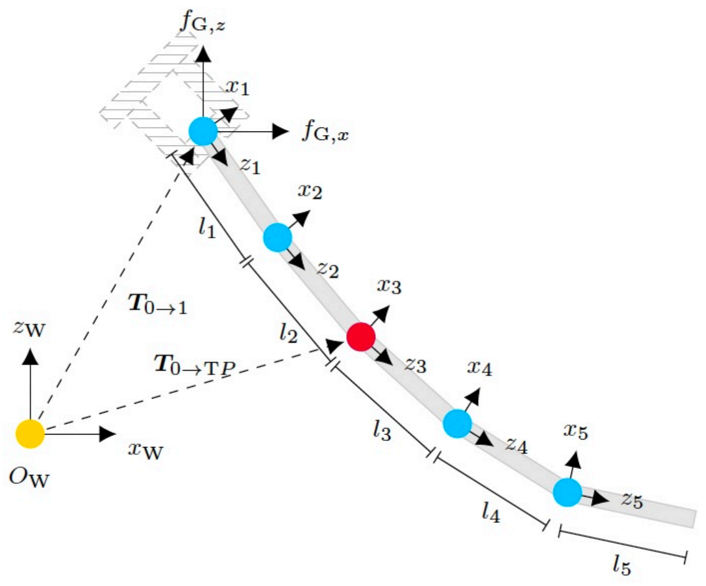


Figure 2.9: 2D schematic representation of the kinematics of the DLO with COSs being located in the joints of the multibody system.

the number of segments used. A balance is sought between a sufficient number of segments to accurately represent the DLO's shape and the computational complexity of the model.

This approach has a balanced between physical accuracy and computational feasibility, making it practical for real-time use. It provides flexibility through rotational degrees of freedom and realistic physical behavior with friction and damping forces. However, as you add more segments to make the model more accurate, it becomes more computationally demanding, which can slow things down in real-time applications.

**Discrete Elastic Rod Model**

For proving a detailed and robust representation of the DLO's behavior under various manipulative forces, which is essential for developing precise control strategies in robotics, a model is chosen. The discrete elastic rod model, a sophisticated framework that integrates the fundamental physical properties of DLOs—stretching, bending, and twisting deformations that is represented in the paper [16].

The DLO is conceptualized as a series of connected cylindrical segments (or an elastic rod), where each segment's dynamics can be modeled in terms of stretching, bending, and twisting forces. This discrete representation allows for capturing the complex behaviors of DLOs under manipulations.

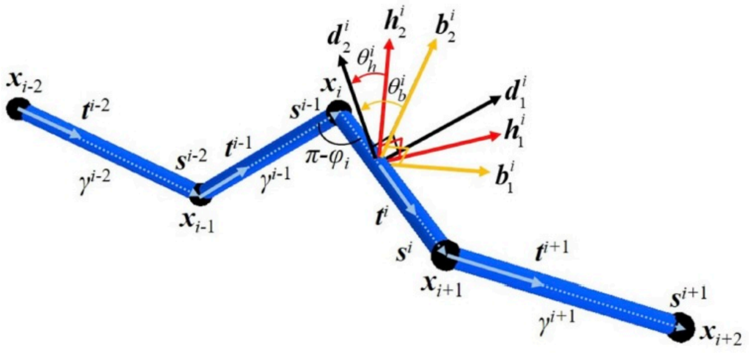


Figure 2.10: Discrete elastic rod model.

The total elastic potential energy $\Pi$ of the rod is expressed as the sum of stretching $\Pi_s$, bending $\Pi_b$, and twisting $\Pi_t$ potential energies:

$$\Pi = \Pi_s + \Pi_b + \Pi_t \tag{2.23}$$

Stretching energy $\Pi_s$ accounts for the elongation or compression of the rod and is defined as:

$$\Pi_s = \frac{1}{2} \sum_{i=0}^{n} EA \left( \frac{|s_i| - |s_i|}{|s_i|} \right)^2 \tag{2.24}$$

Where $EA$ is the stretching stiffness, $|s_i|$ is the current length of segment $i$, and $|s_i|$ is its initial length.

Bending energy $\Pi_b$ is related to the rod's bending deformation and is expressed as:

$$\Pi_b = \frac{1}{2} \sum_{i=1}^{n} \frac{EbI}{\bar{l}_i} (\kappa_i - \bar{\kappa}_i)^2 \tag{2.25}$$

Here, $EbI$ is the bending stiffness, $\kappa_i$ is the curvature of segment $i$, and $\bar{l}_i$ is the average length of the segments around point $i$.

Twisting energy $\Pi_t$ reflects the rod's resistance to torsional deformation:

$$\Pi_t = \frac{1}{2}\sum_{i=0}^{n} \frac{GJ}{\bar{l}_i}(\theta_i - \overline{\theta}_i)^2 \tag{2.26}$$

Where $GJ$ represents the twisting stiffness, and $\theta_i$ denotes the twist angle between adjacent segments.

The discrete elastic rod model used for modeling deformable linear objects (DLOs) offers a comprehensive representation of physical properties such as stretching, bending, and twisting deformations, which is crucial for precise control strategies. It accurately reflects real-world behaviors and includes interactions with the environment, like collisions and friction, validated through experiments and simulations. However, it can be quite complex and computationally demanding. The model also relies heavily on accurately determined parameters and can be challenging to scale up for larger systems or more complex tasks.

**Cosserat Rod Model**

Discretization method that cable is devided into multiple articulated rigid cylindrical-link segments is another modeling approach for long flexible cables. It enables the simulation of complex behaviors such as winding, knot tying, and lifting, which are pivotal in both industrial and surgical robotics, it is introduced in the paper [12].

The modeling of the deformable object leverages the Cosserat rod theory, which provides a framework for representing the deformation and dynamics of slender, rod-like objects. The theory allows for the consideration of extension, shear, bending, and torsion effects between linked segments.

The Cosserat rod model is employed to calculate the strains along the cable's centerline using the equations:

- Extension and Shear Strain:
$$\Gamma(s) = \delta r(s) - d_3(s) \tag{2.27}$$
- Bending and Torsion Strain:
$$\Omega(s) = 2\bar{q}^*(s)\frac{\partial}{\partial s}\bar{q}(s) \tag{2.28}$$

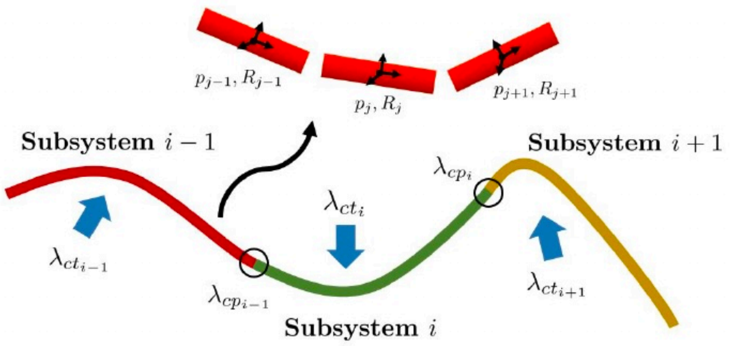


Figure 2.11: Discretization structure of the cable

where $r(s)$ represents the centerline of the rod, $d_i$ are the local directors (frame vectors), $\Gamma(s)$ and $\Omega(s)$ are the strains representing extension/shear and bending/torsion respectively, and $\bar{q}(s)$ is the orientation of the segment represented in quaternion form.

Segmentation and Dynamics Formulation:

The discrete strain energy (denoting the deformation energy within each segment and between adjacent segments) is formulated to capture all possible motions of the cable. The strain energies due to extension/shear and bending/torsion are captured and represented as follows:

$$\psi_e = \frac{1}{2} e_1^T [K_1] e_1 + \frac{1}{2} e_2^T [K_2] e_2 = \frac{1}{2} e^T [K_e] e \tag{2.29}$$

This formula encapsulates the strain energy where $e_1, e_2$ are the constraint errors for extension/shear and bending/torsion, respectively, and $[K_1]$, $[K_2]$ are the gain matrices derived from the Cosserat model, influencing the stiffness of the cable segments in response to deformation.

This modeling approach includes real-time performance, physical accuracy, and the ability of complex behaviors such as winding, knot tying, and lifting. The use of Cosserat rod theory and Passive Midpoint Integration (PMI) ensures stability and accurate representation of extension, shear, bending, and torsion effects. However, this method also has challenges, such as high computational complexity, the need for careful initialization and parameter tuning, and reliance on certain simplifications. Handling complex contact scenarios can be difficult.

**Multivariate Dynamic Splines**

For modeling of DLOs a strategy to approximate the complex, dynamic behavior of DLOs is based on Multivariate Dynamic Splines, introduced in the paper [21]. The DLO is represented through spline basis, which is a function of a free coordinate $u$ running along the cable's length from one end ($u = 0$) to the other ($u = L$), where $L$ is the total length of the cable. This representation is chosen for its effectiveness in computing the shape and spatial derivatives of the DLO straightforwardly, and for its property of minimizing the model curvature, reflecting the physical behavior of DLOs.

The configuration of the DLO at any point along its length is given by the function $q(u)$, which includes both linear coordinates $x(u), y(u), z(u)$ and the axial DLO twisting $\theta(u)$:

$$q(u) = \sum_{i=1}^{n_u} b_i(u) q_i \tag{2.30}$$

where $q_i$ are the control points that interpolate the DLO shape through the spline polynomial basis functions $b_i(u)$, and $n_u$ is the number of control points.

The dynamic behavior of the DLO is described by applying the Lagrange equations to the system, which involve the calculation of kinetic and potential energies as well as the external forces acting on the DLO.

Using a spline basis to model the DLO allows for efficient computation of the shape and its spatial derivatives that is crucial for real-time applications. The spline model minimizes the curvature, this ensures that the model behaves realistically under various manipulations. Moreover, the iterative solution framework ensures quick convergence and robustness to different initial conditions and external forces.

## 2.2 Planar

Planar deformable objects are two-dimensional surfaces that can undergo continuous deformations such as bending, stretching, and twisting without losing their essential properties or structure. The problem of manipulating highly deformable materials such as clothes and fabrics frequently arises in different applications. These include laundry folding, robot-assisted dressing or household chores, ironing, coat checking, sewing, and transporting large materials like cloth, leather, and composite materials

### 2.2.1 Numerical

For modeling the behavior of cloth-like materials during manipulation an approach is presented in the paper [10] a discrete model used to simulate the dynamics and interaction of cloth with both human and robot's end effectors. This model allows for a detailed representation of the cloth's behavior under manipulation.
The cloth is discretized into a set of nodes, denoted as $C$, where each node $i \in C$ has a position $p_i$ and is connected to a surrounding neighborhood of nodes $N_i$, forming a network that represents the cloth.
The set of all nodes $C$ is partitioned into a subset of free nodes $F$ that are not directly grasped but are influenced by the overall cloth dynamics and external forces like gravity and a subset of grasped nodes $G = \{G_{Rl}, G_{Rr}, G_{Hl}, G_{Hr}\}$.

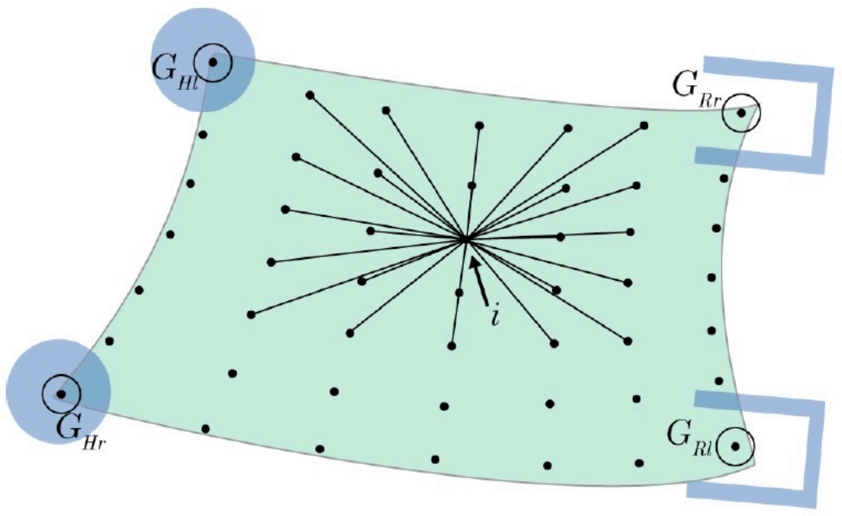


Figure 2.12: The model representation for the discretized cloth

The grasped nodes are attached to either the robot's end effectors ($G_{Rl}$ and $G_{Rr}$ for the left and right, respectively) or the human's hands ($G_{Hl}$ and $G_{Hr}$ for the left and right, respectively).

These grasped nodes are considered rigidly connected to the entity (robot or human) manipulating them, forming boundary conditions for the movement of the free nodes $F$. These free nodes move according to the forces exerted on them, which include external forces such as gravity or drag, and internal forces arising from the inter-node connections.

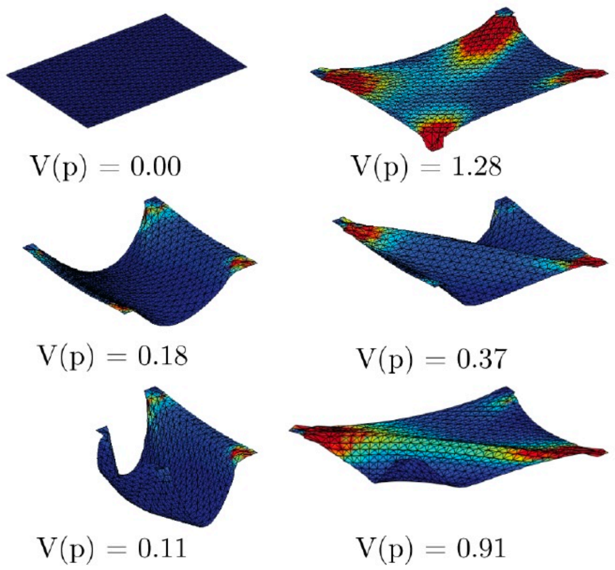


Figure 2.13: Examples of cloth model in different configurations, colored by individual node costs. Note that V(p) = 0 is unattainable due to gravity.

The cloth's behavior is governed by the interactions between these nodes. The forces between the nodes are modeled using a spring-like behavior, where the potential energy $V(p)$ due to deformation from the rest state is calculated using the following spring energy equation:

$$V(p) = \sum_{i \in C} \sum_{j \in N_i} \left( \|p_i - p_j\| - \ell_{ij} \right)^2 \tag{2.31}$$

Where: $p$ is a vector representing the positions of all nodes in set $C$, $N_i$ is the set of neighboring nodes connected to node $i$, $\|p_i - p_j\|$ is the current distance between nodes $i$ and $j$, $\ell_{ij}$ is the natural (rest) length of the edge connecting nodes $i$ and $j$.

This formulation helps in understanding how the cloth will react to being manipulated, by calculating the potential energy which in turn is minimized during the simulation to ensure that the cloth remains as close to its natural state as possible while being manipulated.

The discrete node-based model, gives a detailed and realistic way to simulate how cloth behaves, by providing a detailed and realistic simulation, this modeling approach allows for precise control and adjustment during manipulation tasks, making it highly suitable for collaborative scenarios where human intuition and robotic precision need to work together seamlessly. making it good for human-robot collaboration. It uses a network of nodes to represent the cloth, allowing the robot to move accurately based on human input. However, this method can be very complex and computationally expensive. It needs careful setup and fine-tuning, and the robot's movement limitations can make control tricky. The model reacts to human actions rather than predicting them, which can sometimes lead to less efficient handling.

**Mass-Spring Model**
In human-robot co-manipulation, precise modeling of deformable objects such as fabrics is critical for achieving effective and realistic interaction dynamics. A mass-spring model to represent the complex behaviors of fabric materials is utilized in Dynamic Fabric Simulator (DFS) that is introduced in the paper [18]. This model features a grid of interconnected point masses, linked by various types of non-linear, massless springs designed to capture the unique mechanical properties of fabric, including tensile strength, shear deformity, and bending resistance.

The core of the DFS is a mass-spring model, where the fabric is represented as a network of point masses. These masses are interconnected through three types of non-linear, massless springs, which simulate the fabric's inherent mechanical properties. The springs are categorized based on their function: to provide structural integrity, simulate shear deformation, and mimic bending resistance.

Structural Springs: Connect each mass to its immediate neighbors in the grid, providing the primary mechanism for simulating the fabric's tensile strength and elasticity. Shear Springs: Link masses diagonally across the grid squares. These springs allow the model to simulate the shear behavior of the fabric, which is crucial for representing how the material deforms under angular forces. Bending Springs: Extend beyond immediate neighbors to include second or third nearest masses. Bending springs are crucial for modeling how the fabric bends and folds, contributing to a more realistic simulation of complex deformations.

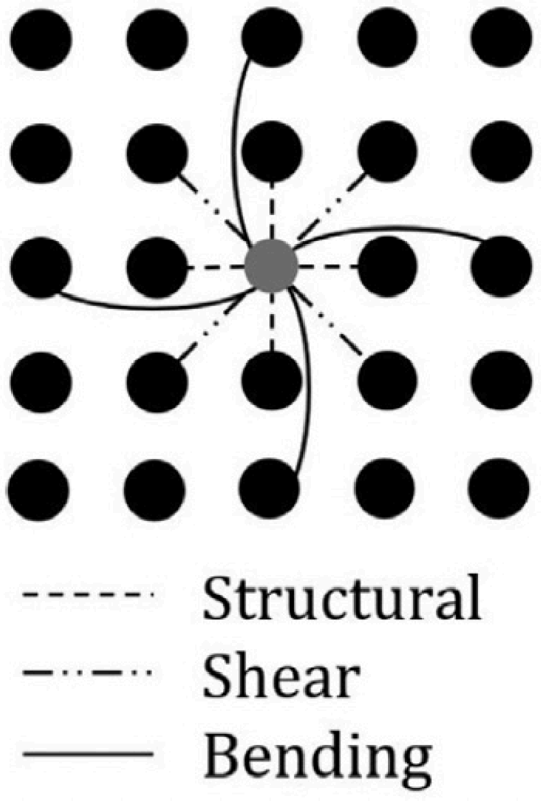


Figure 2.14: Modelling

The forces exerted by the springs are governed by polynomial equations to reflect mechanical properties. Verlet integration is employed to compute the movement of each mass in the grid, providing stability and simplicity by focusing on energy conservation. The model parameters, including spring constants, mass values, and damping coefficients, are adjustable to match various fabric types, with initial values derived from empirical data or experimentation to ensure realistic simulation under different external forces such as gravity and manipulation.

The mass-spring model-based approach for human-robot co-manipulation of deformable objects provides a realistic simulation of fabric behavior by accurately capturing mechanical properties such as tensile strength, shear deformity, and bending resistance. Moreover, the approach minimizes physical damage to fabrics and is widely applicable due to its compatibility with various human tracking systems and robot brands. However, this method also has some drawbacks. It requires precise and sometimes complex calibration of parameters, which can be time-consuming. The computational load, despite being managed by parallelization, may still be significant, and the high parametrization and applicability, real-time performance through parallelization, and seamless integration with human actions and system reconfiguration makes the this approach more suitable for industrial applications.

**Mass-Spring Model**

Mass spring model that is used in the paper [7], essential for simulating the physical properties and behaviors of textiles and similar materials. It is able to to represent complex deformable objects as two-dimensional meshes within a three-dimensional simulation environment, facilitating a detailed analysis of material behavior under various manipulative forces.
The choice of the mass-spring model is advantageous because it allows for the adjustment of mechanical properties such as mass, elasticity, and damping, which are critical for capturing the dynamic responses of flexible objects to external forces like gravity and robotic handling.

The mass-spring model is central to this approach, abstracting the deformable object into a network of interconnected particles or nodes. These nodes possess inherent mechanical properties, including mass, position, velocity, and acceleration, which are crucial for simulating physical behaviors.

The movement and interaction of each particle within the mesh are governed by Newton's second law of motion, formalized as:

$$f_n = m_n a_n \tag{2.32}$$

where $f_n$, $m_n$, and $a_n$ represent the force, mass, and acceleration of the $n^{th}$ particle, respectively.

The force acting on each particle ($f_n$) comprises both external forces (such as gravity and wind) and internal forces emanating from the springs connecting

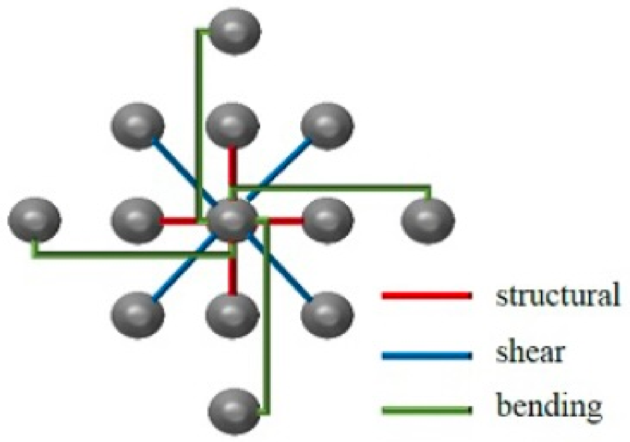


Figure 2.15: Structure of mass-spring model

the particle to its neighbors. These internal forces are summed up as follows:

$$f_{internal} = f_{structural} + f_{shear} + f_{bending} \tag{2.33}$$

where each component represents the force contributions from structural, shear, and bending springs.

Utilizing Hooke's law, the force exerted by each spring is described by:

$$f_n = -k_s(|l_n| - l_0) + k_d \frac{v_n}{|l_n|} \tag{2.34}$$

Here, $l_0$ and $|l_n|$ denote the original and post-deformation lengths of the spring, $k_s$ is the spring's stiffness coefficient, $k_d$ signifies the damping ratio, and $v_n$ is the velocity difference between the spring's endpoints.

To simulate the deformation over time, the explicit Euler integration method is applied, yielding the following relations for updating velocity and position:

$$v_{i,t+1} = v_{i,t} + \frac{f_{i,t}}{m}\Delta t \tag{2.35}$$

$$x_{i,t+1} = x_{i,t} + v_{i,t+1}\Delta t \tag{2.36}$$

where $x_{i,t}$ and $v_{i,t}$ are the position and velocity of particle $i$ at time $t$, respectively, and $\Delta t$ represents the integration time step.

It has several important aspects of using the mass-spring model for simulating deformable objects, such as simplicity and computational efficiency, flexibility in adjusting mechanical properties, and integration with robotic systems for tasks like hanging and folding. However, it has some downsides. It needs careful tuning of parameters to work correctly, and it can become unstable with the wrong settings. The model simplifies the real object's physical characteristics, ignoring things like thickness, and handling collisions accurately is difficult. It might not capture all the complex behaviors of real materials.

### 2.2.2 Data Driven

**Graph Dynamics Modeling**

Modeling deformable objects like cloths poses a challenge due to their complex, high-dimensional dynamics.An approach for a sophisticated understanding and modeling of the complex behavior of cloth deformable objects represented in the paper [15], facilitating their manipulation in robotic applications. It uses graphs and latent representations to understand and predict the behavior of such objects in 3D. By learning from interaction data, this method provides a flexible, generalizable solution for navigating the complicated dynamics of deformable materials.

Graph-Based Representation of Deformable Objects:

The deformable object is modeled as a graph $G_i = (V_i, E_i)$, where $V_i$ represents the nodes, and $E_i$ denotes the edges of the graph. Each node in $V_i$ corresponds to a discrete point on the object, capturing its 3D Cartesian position. The edges in $E_i$ reflect the interaction properties among these points, effectively modeling the structural and elastic relationships that dictate how the object deforms under various forces.

Latent Representation of Physical Properties:

To capture the physical properties of the deformable object, particularly its elasticity, the paper introduces an adaptation module designed to extract a latent representation $z_i$ from observations $O_i$. These observations are derived from interactions with the object, such as pulling, which induce deformations that are indicative of its physical properties. The latent representation $z_i$ is formulated as:

$$z_i = f_\phi(O_i, z_0) \tag{2.37}$$

In this equation, $f_\phi$ is a function that processes the sequence of observations $O_i$, alongside an initial representation $z_0$, to produce the latent representation $z_i$. This latent representation serves as a compact encoding of the object's elastic properties, abstracted from the direct observation of its response to manipulations.

The modeling of deformable objects in this paper through graph dynamics and latent representations enables the prediction of future states of such objects under different interactions, relying on the inherent properties encoded within $z_i$.

The EDO-Net method for modeling flexible objects like cloth has several benefits. It can work with unknown physical properties without needing exact labels, creates a simpler internal representation of the object's properties, and uses graphs to model complex interactions. However, it also has some drawbacks. It needs a lot of computing power, high-quality interaction data, and building accurate graph representations can be complicated and error-prone due to noisy data. The need for sequential observations limits real-time use, and getting complete observations in messy environments can be hard.

**Convolutional Neural Networks (CNNs)**
An adopted method for modeling of the fabric-like object, a carbon fiber ply, relies on a data-driven approach utilizing depth map feedback from a 3D camera, and not rely on traditional physics-based models of material behavior. It is The integration of a RGB-D camera for depth map feedback and the application of Convolutional Neural Networks (CNNs) that enables precise and dynamic manipulation of deformable materials. This technique is introduced in the paper [20]

The material is approximated as a membrane, meaning it's considered to lack flexural rigidity and incapable of sustaining compressive loads. This is a crucial assumption since it simplifies the material's behavior to primarily being affected by traction forces and displacements.

A RGB-D camera captures the depth map of the material, which is used to observe the shape and deflections of the material under manipulation. This visual feedback serves as the basis for understanding the material's deformation without a physical model.

The CNNs processes the depth images to estimate the deformation of the material. The networks learn the relationship between the visual deformation observed in the depth images and the mechanical status (displacements and deformations) of the material.

It allows for real-time performance and computational efficiency, simplifying modeling by focusing on traction forces and displacements without requiring complex physics-based models. This method is flexible, adaptable, and generalizable to various materials, providing accurate deformation estimation. However, it has some drawbacks, including dependency on high-quality training data and limited physical insights due to its black-box nature. The approach currently handles only deformations in principal directions and is sensitive to visual noise and environmental conditions, such as lighting and material surface properties. Despite these limitations, it remains an efficient and practical solution for dynamic material manipulation scenarios.

**G-DOOM (Graph dynamics for Deformable Object Manipulation)**
A model that uses a graph-based framework to abstract deformable objects as interacting sets of keypoints, allows for a detailed and abstract modeling of object dynamics, introduced in the paper [17]. G-DOOM (Graph dynamics for Deformable Object Manipulation). This framework captures the complex dynamics of deformable objects using a combination of unsupervised learning for keypoint extraction and graph neural networks (GNNs) to model the interactions between these keypoints.

Keypoints are extracted from depth images using unsupervised learning, which identifies visually prominent features critical for understanding the object's dynamics.

Feature extraction and keypoint detection process involves a feature extractor $f_{\text{enc}}$ and a keypoint detector $f_{\text{kp}}$, which produce image features $\phi_{\text{src}}, \phi_{\text{tgt}}$ and keypoint heatmaps $H_{\text{src}}, H_{\text{tgt}}$ respectively. For reconstruction loss the system optimizes the keypoint detection by minimizing the reconstruction loss $L_{\text{rec}}$, which is defined as the error between the target image $I_{\text{tgt}}$ and its reconstruction $\hat{I}_{\text{tgt}}$.

The interactions between keypoints are modeled using a graph neural network, where keypoints are nodes in the graph.

For graph construction, a graph $G_t$ at time $t$ is defined with nodes $V_t$ representing the keypoint features $v_i^t$ and edges $E_t$ denoting the interactions among keypoints. Then for node features, each node feature $v_i^t$ integrates depth and geometric information pertinent to the keypoint's position.

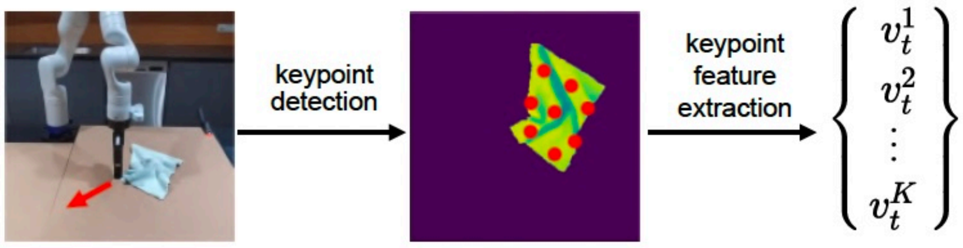


Figure 2.16: keypoint detection, extracts the corresponding keypoint features

The dynamic behavior of deformable objects is modeled over time using recurrent graph dynamics, adapting to changes and handling partial observability.

The features of each node are updated by a GNN based on the current features $v_i^t$, the previous hidden state $h_t$, and the current action $a_t$:

$$v_i'^t = GNN([v_i^t, h_t, a_t], Nb(v_i^t)) \tag{2.38}$$

where $Nb(v_i^t)$ are the neighboring nodes of $v_i^t$ in the graph. In hidden state update, the global state or belief about the object $h_{t+1}$ is updated using an RNN:

$$h_{t+1} = RNN(h_t, Pool(\{v_i'^t\})) \tag{2.39}$$

The G-DOOM framework uses keypoints to represent objects, making the model detailed efficient. Unsupervised keypoint extraction make robustness and adaptability allowing the model to adapt to different shapes and configurations. Graph neural networks (GNNs) help to model the interactions between these keypoints accurately. The model also uses recurrent neural networks (RNNs) to keep track of changes over time, making it good at handling dynamic objects. Additionally, it works well both in simulations and real-world tasks like rope straightening and cloth manipulation. However, the model's performance depends on correctly identifying keypoints, and it may struggle with more complex tasks that require a detailed understanding of 3D shapes. Training the model is complex and requires a lot of data.

## 2.3 Three-Dimensional

### 2.3.1 Numerical

**Combining mesh and volumetric grid structures**
This model includes a mesh structure for immediate reference and a volumetric grid for ongoing reconstruction, which allows for parallel processing and efficient shape adaptation. This method leverages the strengths of traditional model-based methods while enhancing adaptability and speed in real-time applications that is represented in the paper [6].

Encoding the surface geometry and texture into a 3D volumetric grid structure is more feasible since the shape reconstruction progress can be implemented efficiently via parallel operation on the grids. To combine the advantages of both representation, by projecting multiple image frames data back into the space of a reference frame $F^0$ (which is usually set as the initial frame) according to the estimated interframe deformations and then integrating these frames into a reference mesh $M^0$. For efficient image integration, the reference mesh $M^0$ is maintained via a discrete truncated signed distance function (TSDF) volume, (as illustrated in Figure 2.18).

which we denote as the reference volume $V$. In this reference volume $V$, the surface geometry is voxelized as $f D(x^0) f D(\Omega^0)$, where
$D(x^0) \in [-1, +1]$ encodes the truncated signed distance value for each voxel $x^0$,

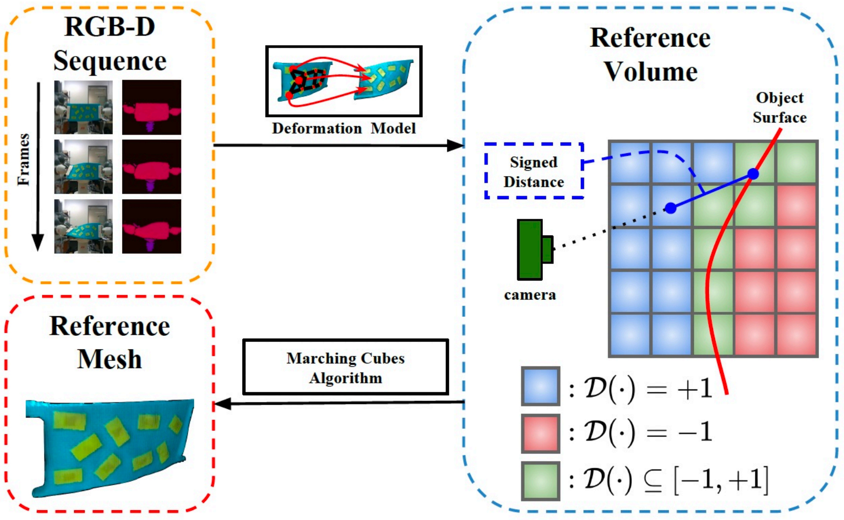


Figure 2.17: Reference volume V in 2D case.

and $\Omega\ (x^0) \in [0, 1]$ is the associated weight. The content of each voxel will be updated independently for image integration. To obtain a high-quality mesh with texture, we also maintain the RGB information $C(x^0) \in [0, 255]^3$ in each voxel. As a summary, our reference volume can be represented as

$$V(x^0) = \{D(x^0), C(x^0), \Omega(x^0)\} \tag{2.40}$$

From the reference volume $V$, we extract the reference mesh $M^0$ with Marching Cubes algorithm. The reference mesh $M^0$ can be further deformed according to the estimated inter-frame deformation model to obtain the live mesh $M^t$, which indicates the object shape in the live frame $F^t$. For the convenience of discussion, we define the mesh vertices and corresponding normals as $(v_m, n_m)_{m=1}^{M}$ in this work, where $M$ is the number of vertices in the mesh.

This way of modeling of deformable object includes real-time processing crucial for dynamic environments, a model-free design increasing adaptability to various materials and shapes, and robustness to noise and occlusion ensuring reliable operation in real-world conditions. Combining mesh and volumetric grid structures used the strengths of both for efficient shape adaptation and high-quality surface reconstruction. However, the model faces limitations such as error accumulation in deformation estimation, challenges in handling large deformations and topological changes (e.g.tearing, folding in complex ways), and high computational demands requiring powerful hardware.

**FEM (Finite Element Method)**
A model-based approach using the Finite Element Method (FEM) that introduced in the paper [3] to enable the dexterous handling of such objects by robotic systems. With using a volumetric linear FEM. It represent how FEM is utilized to create a detailed and responsive model that adjusts to dynamic manipulative forces.
The deformable object is modeled using a volumetric linear Finite Element Method (FEM), which is well-suited for representing continuous isotropic materials. This modeling approach involves tessellating the object's volume into a mesh of elements, specifically tetrahedrons, that connect a set of 3D vertices. The manipulation of the object is then formulated as a quasi-static equilibrium between the internal forces within the deformable structure, $f(x)$, and the external forces, such as those exerted by the robot hand at the contact points.

Mathematically, the internal forces $f$ in the object are balanced by the external forces, with a focus on the forces applied at the fingertips of the robotic hand, assuming these are the primary points of interaction. The relationship between the internal forces $f$ and the displacements $\hat{u}$ of the mesh vertices is governed by Hooke's law and the infinitesimal strain theory, and can be expressed as:

$$f = K\hat{u} \tag{2.41}$$

where $K$ is the stiffness matrix, which depends on the material's elastic properties, specifically the Young's modulus $E$ and Poisson's ratio $\nu$. This linear relation is adapted for rotational transformations through a corotational approach, allowing the model to accommodate for rotations by warping the stiffness matrix with respect to a rotation matrix that represents the rotational component of the element deformations.

To estimate the material's elasticity parameters $E$ and $\nu$, the paper outlines a minimization strategy based on fitting the simulated deformations—obtained from the FEM model under applied forces—to the observed deformations captured with an RGB-D sensor. In the simulation setup, a force measured by a force sensor on a robotic arm is applied to the object, and the resulting deformation is captured. The fitting error between these simulated and observed deformations is then minimized to effectively estimate the elasticity parameters.

This FEM-based method provides high precision and accuracy by using detailed modeling and real-time adjustments, making it suitable for handling deformable objects. It has consistency with physical laws and flexibility, allowing for precise control of deformations through accurate elasticity parameters and dynamic responses to forces. However, it faces challenges such as high computational demands for real-time FEM calculations, reliance on precise parameter estimation.

**Growing Neural Gas (GNG) and Particle Graph Networks (PGN)**
The manipulation of non-rigid or deformable objects presents a unique set of challenges, specially when it comes to the real-time tracking and modeling necessary for effective interaction. Two sophisticated graph-based architectures, Growing Neural Gas (GNG) and Particle Graph Networks (PGN), which are pivotal for understanding and predicting the behavior of deformable objects during manipulation introduced in the paper [25].

GNG is a type of unsupervised learning algorithm that learns a topological representation of a data distribution as a dynamic undirected graph $G = \langle A, N \rangle$, where $A = \{a_i\}$ is the set of nodes (or neurons) and $N = \{n_k\}$ is the set of edges. Each node $a_i = \{w_i\}$ in the graph has an associated weight vector $w_i \in \mathbb{R}^n$, which is of the same dimension as the input space $n$. The network adapts over time by incrementally adding nodes and modifying edges based on criteria like threshold distances, effectively capturing the structure of complex changing data distributions.
The paper it is said that while GNG can represent changing structures efficiently and robustly against noisy data, it struggles with the real-time processing demands required for robotic manipulation due to computational limitations.

PGN is a representation inspired by physics simulation models, where entities are represented as graph neural networks, enabling the prediction of system dynamics by learning parameters from data. PGN is defined as a directed graph $G = \langle O, R \rangle$, with $O = \{o_i\}$ representing the nodes (or objects) and $R = \{r_k\}$ representing the edges (or relations).

Each node $o_i$ is described by $o_i = \{x_i, a_{o_i}\}$, where $x_i = \{q_i, \dot{q}_i\}$ represents the state (position, velocity) and $a_{o_i}$ represents features (stiffness, radius). Relations

$r_k$ are described by

$$r_k = \langle u_k, v_k, a_{r_k} \rangle \tag{2.42}$$

capturing the features of the relation between nodes.
The model predicts the next state of each object

$$o_{i,t+1} = f_O(o_{i,t}, e_{k,t}) \tag{2.43}$$

and the effect of each relation

$$e_{k,t+1} = f_R(o_{u_k,t}, o_{v_k,t}, a_{r_k}) \tag{2.44}$$

with $f_O$ and $f_R$ being functions approximated using neural networks.

GNG could provide a robust structure representation by efficiently capturing and representing the shape and deformation of objects from sensor data. This structured representation could then be input into PGN, which would predict how the object's shape changes over time, facilitating real-time manipulation.

The approach using Growing Neural Gas (GNG) and Particle Graph Networks (PGN) to model deformable objects is strong in several ways. GNG is good at capturing the shapes and changes of objects, even when the data is noisy, and PGN predicts how these shapes will change over time, helping robots manipulate objects more accurately. However, GNG needs a lot of computing power, making it hard to use in real-time, combining these models is also technically challenging and requires fine-tuning.

**Corotational FEM**
A corotational FEM variant, the method enhances its ability to handle significant rotational distortions without sacrificing accuracy, a sophisticated computational technique renowned for its precision in simulating and analyzing complex deformations for realistic modeling of objects subjected to diverse and unpredictable interactions that is discussed in the Paper [22], introduced.

The deformable object is represented through tetrahedral meshes. The relationship between stress and strain in the object, due to deformation, is described using Hooke's law. This law is applicable for small deformations of linear elastic objects and relates stress to strain through a linear equation dependent on the object's elasticity parameters, namely the Young's modulus and the Poisson's ratio.

The linear algebraic equation of motion for the deformable object is given by:

$$M\frac{d^2}{dt^2}(x(t) - xu) + D\frac{d}{dt}(x(t) - xu) + K(x(t) - xu) = f_{\text{ext}} + f_{\text{int}} \tag{2.45}$$

$n$ is the number of vertices of the mesh, $M \in \mathbb{R}^{3n\times 3n}$ is the mass matrix, $D \in \mathbb{R}^{3n\times 3n}$ is the damping matrix, $K \in \mathbb{R}^{3n\times 3n}$ is the stiffness matrix (depending on the Young's modulus and Poisson ratio), $x(t) \in \mathbb{R}^{3n}$ (resp. $xu \in \mathbb{R}^{3n}$) is the position of each vertex of the mesh at time $t$ (resp. in the undeformed state), $f_{\text{ext}} \in \mathbb{R}^{3n}$ is the external forces applied, $f_{\text{int}} \in \mathbb{R}^{3n}$ is the internal forces resulting from the deformation.

It handles rotational deformations accurately due to the corotational FEM, simulates complex deformations precisely with FEM, and works well with various unpredictable forces. Using Hooke's law makes calculations simpler. It does not require external fiducial markers, making setup easier. However, it only works well for small deformations because Hooke's law is for linear elastic objects. It's also computationally demanding and relies on accurate initial models.

### 2.3.2 Analytical

**A Set of Discrete Points**

A modeling that provides a structured way to understand and predict how manipulations of certain points on the object will result in overall deformation, enabling the development of effective control strategies for deformable object manipulation introduced in the paper [8].

A set of discrete points, consisting of manipulated points (pm), feedback points (pf), and uninformative points (pu). The relationship between the feedback points and the manipulated points is modeled mathematically as follows:

$$\delta pm = F(\delta pf) \tag{2.46}$$

where $\delta pf = pf - pf^*$ and $\delta pm = pm - pm^*$ represent the displacements relative to the equilibrium for feedback points and manipulated points, respectively.

To abstract the deformation behavior further and facilitate the learning process, a low-dimensional feature vector $x$ is extracted from the feedback points $pf$, transforming the model into:

$$\delta pm = F(Q^{-1}(pf^*)\delta x) = Z(\delta x) \tag{2.47}$$

where $Q(\cdot)$ is the feature extraction function, and $Z(\cdot)$ is termed the deformation function that the controller aims to learn.

For better illustration, Figure 2 models a soft object using these three classes of points. The manipulated points are those directly controlled by the robot, the feedback points are used for monitoring deformation, and uninformative points do not significantly contribute to the model or control process.

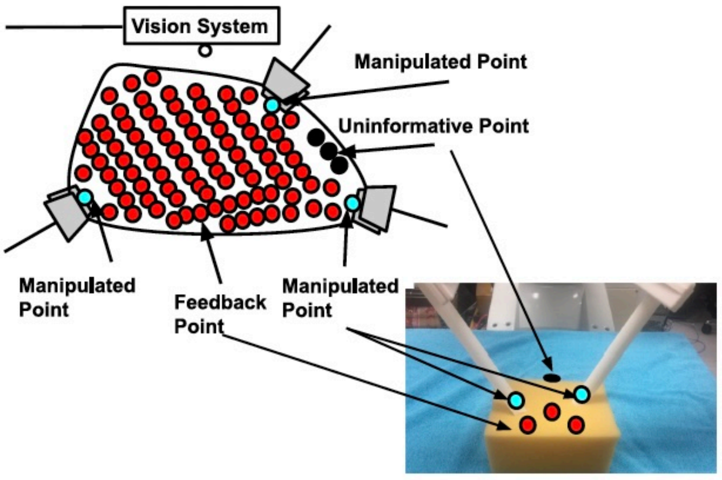


Figure 2.18: soft object modeling.

The modeling approach for deformable objects as a set of discrete points simplifies the application of deep neural networks, facilitating real-time prediction and manipulation by breaking down complex deformations into manageable components. This method benefits from efficient feature extraction, a clear mathematical relationship between feedback and manipulated points. However, it relies heavily on accurate feature extraction, requires initial training data, assumes an equilibrium state, and may oversimplify highly complex deformations. Additionally, scalability issues can arise as the number of discrete points increases, and it can impact the real-time performance.

# 3 Simulation

## 3.0.1 Introduction

Simulations play a crucial role in the field of robotics, particularly in the manipulation of deformable objects like fabrics, cables, and soft tissues. These objects exhibit complex behaviors that are influenced by numerous factors, making their modeling and prediction challenging. Simulations provide a way to understand these behaviors accurately without relying on extensive physical trials. They are invaluable during the design and development stages, allowing engineers to optimize mechanical components and control algorithms efficiently. Additionally, simulations offer a safe, risk-free environment to test and refine robotic operations, ensuring safety and effectiveness when applied in real-world scenarios. This approach is also cost-effective, significantly reducing the expenses and time associated with building physical prototypes and conducting multiple experiments.

Furthermore, simulations are essential for testing and validating models before their real-world implementation. They enable precise control over experimental variables, ensuring accuracy and consistency in results. By allowing repeated experiments under identical conditions, simulations verify the reliability of models. They also facilitate comprehensive scenario testing, including extreme conditions that are difficult to replicate physically, ensuring robustness and versatility of the models. Parameter sensitivity analysis, integration checks, and iterative feedback further refine these models, making them more reliable and effective. Additionally, simulations aid in benchmarking different models to select the best-performing ones and ensure compliance with regulatory standards in fields such as healthcare and automotive.

### 3.0.2 Simulation Frameworks and Techniques

Simulating deformable objects involves using computational frameworks that can accurately model the complex behaviors of these objects under various conditions. These frameworks are essential for developing and validating robotic systems designed to manipulate deformable objects, providing insights into their mechanical properties and dynamic responses. Several key simulation frameworks are used in this domain, including Finite Element Method (FEM), Mass-Spring Models, Particle-Based Simulations, and other relevant techniques.

**Finite Element Method (FEM)**
FEM is a powerful computational technique used to approximate the behavior of deformable objects. It works by breaking down a complex object into smaller, manageable finite elements, such as tetrahedra or hexahedra. Each element is governed by equations derived from the object's material properties and external forces. The process involves discretizing the object into a mesh of finite elements, applying boundary conditions and external forces, and solving the equations iteratively to predict the object's deformation and stress distribution. FEM offers high accuracy and is applicable to a wide range of materials and loading conditions, providing detailed insights into stress, strain, and deformation patterns. However, it is computationally intensive, often requiring high-performance computing resources, and can be time-consuming, making real-time applications challenging.

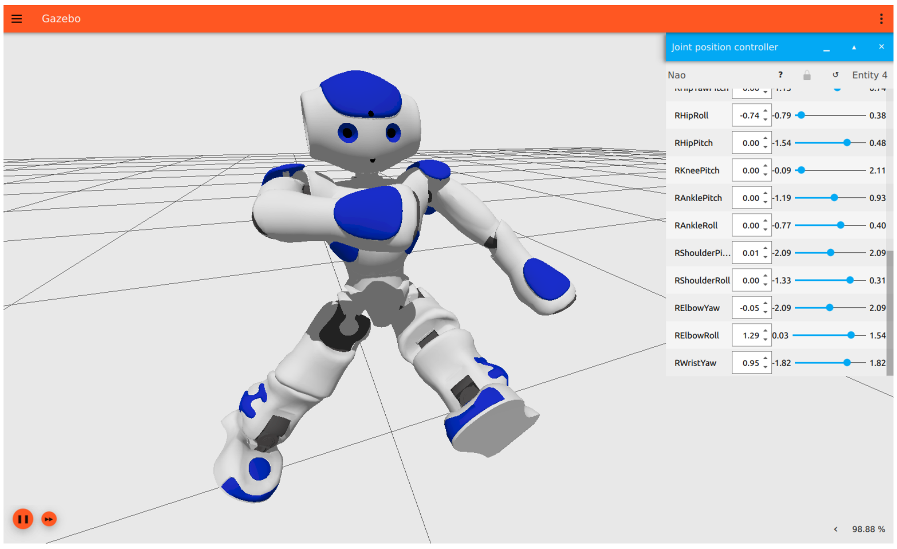
**Mass-Spring Models**
simplify the representation of deformable objects by modeling them as a network of masses connected by springs. This approach is particularly useful for simulating objects like cloth, ropes, and flexible sheets. In this process, the object is represented as a collection of point masses connected by springs, which represent tensile and shear forces. The behavior of each spring is governed by Hooke's Law, and damping factors are included to model energy dissipation. Mass-Spring Models are relatively simple and computationally efficient, making

them suitable for real-time applications. However, they have limited accuracy compared to more detailed methods like FEM and may not capture complex material behaviors and interactions. They require careful tuning of spring constants and damping factors to match physical properties accurately.

**Particle-Based Simulations**
It models deformable objects as collections of interacting particles. Each particle represents a small volume of the material, and interactions between particles are defined by physical laws governing forces such as attraction, repulsion, and damping. Two common types of particle-based simulations are Smoothed Particle Hydrodynamics (SPH), primarily used for fluids but adaptable for soft bodies, and Position-Based Dynamics (PBD), which focuses on maintaining positional constraints suitable for real-time simulations. This process involves distributing particles within the object's volume, calculating forces between particles based on their positions and velocities, and updating particle positions iteratively to simulate deformation. Particle-Based Simulations are flexible and can model a wide range of behaviors, making them suitable for real-time applications with the right optimizations. However, they are computationally demanding for large numbers of particles and require extensive calibration to match physical properties accurately.

**Gazebo Simulator**
It is an open-source multi-robot simulator provides a robust and accurate environment for testing and developing robots in complex indoor and outdoor settings. Gazebo is generally used for simulating rigid body dynamics rather than deformable objects. While there have been some efforts to extend Gazebo's capabilities to handle soft-body or deformable object simulation, it is not its primary strength. While its great features, community support, and integration with ROS make it a powerful tool for many robotics simulation tasks. It is not the best choice for simulating deformable objects requiring high-speed simulation, extreme accuracy, due to its focus on rigid body dynamics.

**Pybullet**
PyBullet is a Python module built on the Bullet Physics Library, designed for real-time simulations with GPU acceleration for better performance in large-scale tasks. This makes it great for robotics, machine learning, and computer graphics. It supports both rigid and soft body dynamics but it is not be the



| | | | |
|---|---|---|---|
| RHipRoll | -0.74 | -0.79 | 0.38 |
| RHipPitch | 0.00 | -1.54 | 0.48 |
| RKneePitch | 0.00 | -0.09 | 2.11 |
| RAnklePitch | 0.00 | -1.19 | 0.93 |
| RAnkleRoll | 0.00 | -0.77 | 0.40 |
| RShoulderPi... | 0.01 | -2.09 | 2.09 |
| RShoulderRoll | 0.00 | -1.33 | 0.31 |
| RElbowYaw | -0.05 | -2.09 | 2.09 |
| RElbowRoll | 1.29 | 0.03 | 1.54 |
| RWristYaw | 0.95 | -1.82 | 1.82 |

Figure 3.1: Gazebo Simulator

best choice for highly detailed deformable object simulations as its accuracy for complex deformable object simulations is limited due to simplifications in its physics models.

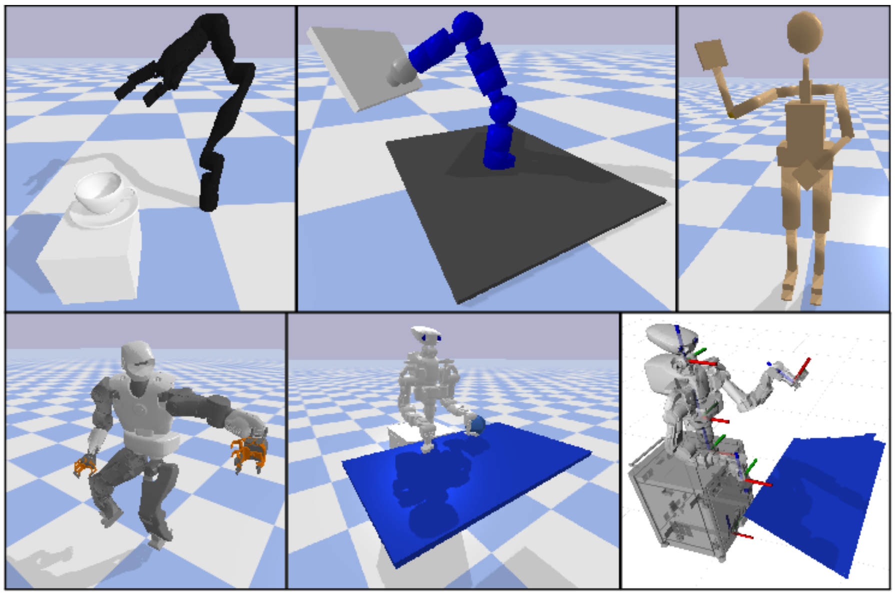


Figure 3.2: Pybullet Simulator

**Unity 3D**

It is a powerful and widely-used game development platform that allows developers to create interactive 2D and 3D content. It has user-friendly interface

making it easy to create and manipulate 3D models. It had a library of pre-made assets and plugins that can speed up the development. Moreover, it is real-time and has an extensive documentation finding resources easily. While Unity is developed for real-time applications, simulations that require highly detailed physics calculations for deformable objects can be computationally intensive and might suffer in performance. It primarily designed for real-time applications and may not provide the high level of accuracy needed for simulating of complex deformable objects.
Unity 3D is best suited for applications that require real-time interactivity and visually engaging simulations, such as games, virtual and augmented reality, prototyping and visualization.

**Isaac Gym(DefGraspSim)**
Isaac Gym, developed by NVIDIA, is a physics simulation environment that integrates GPU-accelerated computation for various robotics and reinforcement learning tasks. Within Isaac Gym, DefGraspSim is a specific component designed to handle simulations involving deformable objects. It is based on FEM. It has good advantages for simulating deformable objects, including high-speed performance and scalability due to GPU acceleration, and advanced physics modeling for accurate simulations of complex interactions. However, its reliance on powerful NVIDIA GPUs and high computational demand can is a limitation.

**NVIDIA Isaac Sim**
NVIDIA Isaac Sim is a powerful simulation platform designed for the development, testing, and deployment of autonomous machines and robots. NVIDIA Isaac Sim is a part of the NVIDIA Isaac SDK, offering a wide range of tools for building robots. Isaac Sim is built on NVIDIA Omniverse, a powerful platform that uses multiple GPUs to run realistic and real-time simulations of robots and their surroundings.
Although it creates very realistic physics and detailed graphics, which are important for building and testing robots that can work on their own, it focuses on rigid body dynamics and its capabilities for simulating deformable objects (soft body physics) are limited.

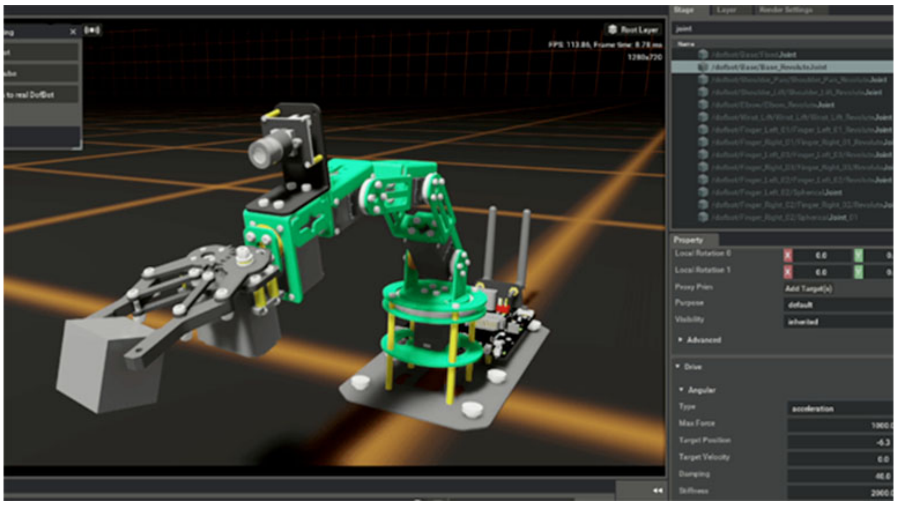


Figure 3.3: Nvidia Issac Sim Simulator

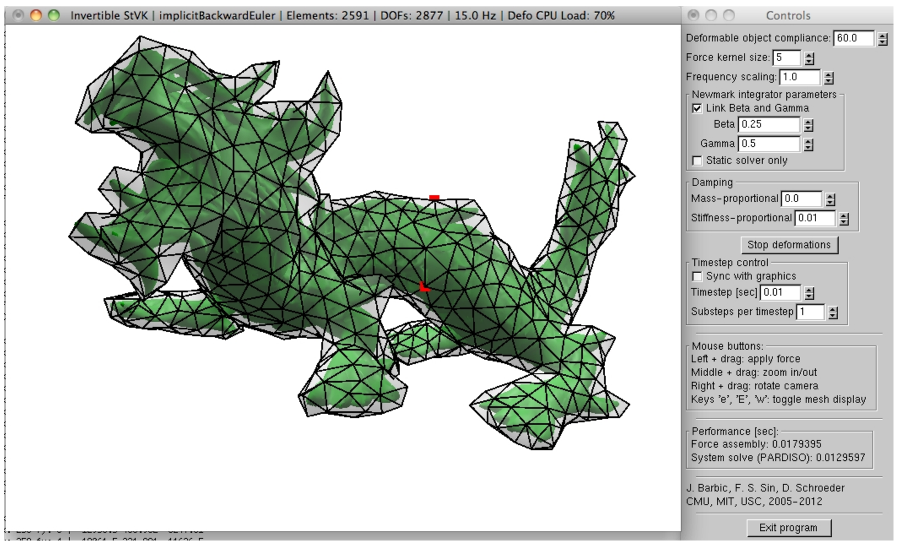
**VegaFEM**

VegaFEM (Vega Finite Element Method) is a computational framework designed for the simulation of deformable objects. It is known for its efficient and accurate finite element method (FEM) simulations. It is open sourc and free that can be customized according to specific needs. It supports different types of simulations including linear and nonlinear elasticity, cloth simulation, and other deformable body dynamics.
Vega is a library for simulation of 3D solids, as well as cloth. The input to Vega is a 3D volumetric mesh with material properties. Vega computes the displacements of mesh vertices at each timestep.
it does not support one-dimensional deformable objects (strands). Vega focuses on basic 3D solid and cloth simulations but lacks support for more complex and dynamic interactions and objects.

**ARCSim: Adaptive Refining and Coarsening Simulator**

ARCSim is a physics-based simulation software specifically designed for simulating the behavior of deformable objects, such as cloth and strands. It has realistic and accurate simulation, it support wide range of deformable object like cloth, hair, soft tissue and other flexible materials, and it is open-source. however, it is computationally expensive and it does not support GPU and it has limited documentation.

**ArtDefo (Accurate Real Time Deformable Objects)**


Invertible StVK | implicitBackwardEuler | Elements: 2591 | DOFs: 2877 | 15.0 Hz | Defo CPU Load: 70%

Controls

Deformable object compliance: 60.0
Force kernel size: 5
Frequency scaling: 1.0
Newmark integrator parameters
☑ Link Beta and Gamma
Beta 0.25
Gamma 0.5
☐ Static solver only
Damping
Mass-proportional 0.0
Stiffness-proportional 0.01
Timestep control
☐ Sync with graphics
Timestep [sec] 0.01
Substeps per timestep 1
Mouse buttons:
Left + drag: apply force
Middle + drag: zoom in/out
Right + drag: rotate camera
Keys 'e', 'E', 'W': toggle mesh display
Performance [sec]:
Force assembly: 0.0179395
System solve (PARDISO): 0.0129597
J. Barbic, F. S. Sin, D. Schroeder
CMU, MIT, USC, 2005-2012

Figure 3.4: VegaFEM Simulator

It is a simulation tool designed for the quick and precise real-time animation of flexible objects. It uses the Boundary Element Method (BEM) to address the computational difficulties associated with simulating deformable materials, making it ideal for use in virtual environments and interactive simulations.
ArtDefo is suitable for simulating flexible objects like cloth and strands. ArtDefo is designed for real-time applications, providing quick updates and low latency that is useful in situations that require quick updates and real-time interactions, such as virtual reality and interactive animations. It has accuracy and it is capturing the necessary details of material behavior. However, demand significant computational power and it does not run on GPU and it is restricts the class of materials which it can handle to those with homogeneous material properties.

**Houdini**

Houdini is a 3D animation software developed by SideFX, widely used in the visual effects, animation, and game industries. Houdini utilizes a node-based procedural approach for creating complex simulations and effects. It offers a variety of solvers and simulation tools for different types of physical phenomena, including: fluid simulations (water, smoke, fire), rigid body dynamics, soft body dynamics, cloth simulations, interactive simulations for real-time effects like games.
Houdini run on GPU. However it Requires significant computational power and the full version of Houdini is quite exppenive.

**SOFA**

SOFA is an open source framework based on the Finite Element Method (FEM), with an emphasis on medical simulation and robotics.
SOFA is real-time and has good documentation. It can run on GPU. However its performance is vary depends om the complexity of the simulation. And for simulation of deformable objects like cloth and also linear deformable objects is not working well. Getting high accuracy in cloth simulation needs detailed models and complex calculations, which can make the simulation slower. Simplifying the models to make the simulation run faster can make it less realistic. Moreover, linear deformation models might not accurately represent how certain materials behave, especially those that show significant non-linear elastic or plastic properties.

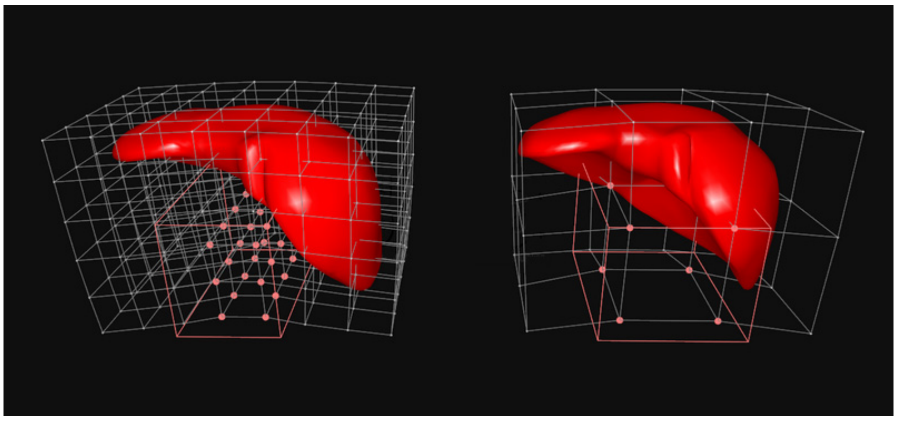


Figure 3.5: Sofa Simulator

**MuJoCo**

MuJoCo is a free and open source physics engine that aims to facilitate research and development in robotics, biomechanics, graphics and animation, and other areas where fast and accurate simulation is needed.
MuJoCo uses a physics-based approach to simulate the motion and interaction of bodies in a three-dimensional space. As MuJoCo has efficient simulation of complex dynamics, high accuracy, run on GPU, and also has a very good documentation, is a good choice for simulation of deformable objects.

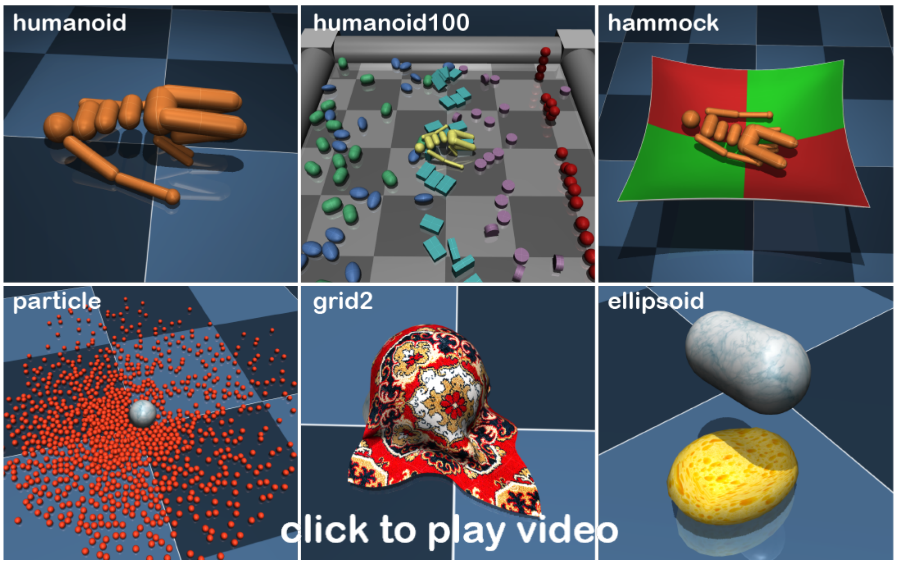


Figure 3.6: MuJoCo Simulator

# 4 Control Strategies

Manipulation of deformable objects is a complex area in robotics that involves handling materials that can change shape, such as fabrics, cables. This presents unique challenges compared to the manipulation of rigid objects, because of high degrees of freedom (DOFs). A control strategy in this context is a set of planned actions and algorithms designed to dynamically manipulate deformable materials to achieve a specific goal, like folding clothes or tying knots. Control strategies are discussed in different kind of deformable objects; linear, planar, and 3D.

## 4.1 Linear

Manipulating linear deformable objects, such as ropes, cables, and wires, is a challenging problem in robotics due to their infinite degrees of freedom and complex dynamics. The control strategies typically focus on shaping these objects into desired configurations while handling their deformable nature.

### 4.1.1 Model-Based Control

Model-based uses mathematical models to predict, analyze, and control system behavior effectively. It mathematically describe the behavior and dynamics of the system under various conditions, forming the basis for designing control laws that can effectively manipulate the system towards desired states.

**Dynamic Regions**

Paper [13] utilized this method to overcome the challenges posed by the high degrees of freedom and coupled interactions along the length of DLOs such as cables or ropes in this paper. Now we are going to look at the specific model-based control strategy in this paper.

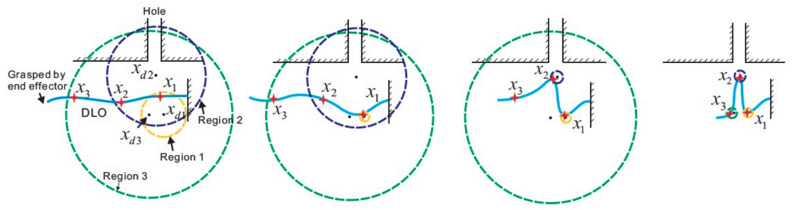


Figure 4.1: An illustration of "Insert-into-Hole"

The strategy utilizes a sequence of dynamic regions for each feature point on the DLO. Each point is allowed to move within a specified region, which adapts dynamically as the manipulation progresses, allowing for flexibility in the control strategy while still guiding the DLO towards the desired shape.
The dynamic region for each feature point is defined mathematically as:

$$f_i(\Delta x_i^T) = \|\Delta x_i^T\|^2 - R_i^2 \tag{4.1}$$

where $\Delta x_i^T$ is the transformed positional error of the ith feature point from its desired position, and $R_i$ is a positive constant defining the initial size of the dynamic region. The control input, adjusting the velocity of the robot's end effector, is given by:

$$u = -\sum_{i=1}^{m} J_i^{-1}(x_i, r) T_i^{-1} (\alpha_i \Delta \epsilon_i + \dot{T}_i \Delta x_i) \tag{4.2}$$

where $J_i(x_i, r)$ is the Jacobian matrix, $T_i$ is a transformation matrix, and $\alpha_i$ are positive constants. $\Delta\epsilon_i$ is the gradient of the potential energy function, driving the feature point back into its region if it exits.

And the potential energy associated with each dynamic region, which influences the control input, is defined to ensure that the feature points remain within their designated dynamic regions:

$$i(\Delta x_i^T) = \frac{k_i}{2}[\max(0, f_i(\Delta x_i^T))]^2 \tag{4.3}$$

And the stability of the closed-loop system, which includes the dynamics of the robotic manipulator and the DLO, is proven using Lyapunov methods. A Lyapunov-like function is constructed from the potential energy functions of all dynamic regions, ensuring that the system's overall energy decreases over time, leading to a stable equilibrium state.

This control strategy has several benefits, such as flexibility and adaptability through dynamic regions, reducing conflicts by controlling movements in sequence, and accurate control. It can also predict how deformable objects will behave during manipulation and is robust in handling coupled movements. Its effectiveness has been proven through experiments. However, there are some challenges, like the complexity and computational demands of mathematical modeling, the need for accurate calibration of the Jacobian matrix, and careful tuning of dynamic region parameters. Although this method avoids conflicts and ensures smoother control, it can be slower and less efficient due to sequential processing.

**Dynamic Surface Control (DSC)**
Dynamic Surface Control (DSC) that is introduced in the paper [2] that is a control strategy for manipulation of deformable linear objects in such a way to suppress the oscillation at the end of the DLO. After modeling of DLO, the first step is decoupling strategy that transforms the system into a form where the actuated and underactuated parts are separated, allowing for the control strategies that specifically target the underactuated dynamics.
After modeling of DLO, the model is transformed into a special cascaded form. This transformation facilitates the separation of the dynamics into actuated and underactuated parts, which is beneficial for applying control laws more effectively (by specifically targeting the underactuated dynamics).

Dynamic Suface Controller (DSC) is designed to suppress oscillations at the end of the DLO. This is designed to ensure that the manipulator brings the DLO back to its origin with minimized vibration, which is critical for ensuring precision in handling deformable objects, ensuring stability and robust performance. This includes two steps:
The first dynamic surface virtual control laws that is based on a combination of state errors and manipulator positioning errors and the actual control law

using these surfaces which integrates the dynamics of the underactuated parts and adjusts them in response to the manipulator's movement and the virtual controls.
DSC strategy is guaranteed through the Lyapunov stability. That is ensuring that all states of the system remain bounded and converge to desired trajectories, indicating that the control strategy is effective in suppressing unwanted swings.

This control strategy for handling deformable linear objects (DLOs) is effective in reducing oscillations, ensuring smooth and precise operations. By strategy separates the system into actuated and underactuated parts, and uses Lyapunov stability criteria to make sure all states remain bounded and converge to desired trajectories. However, the strategy is complex to implement due to the need for accurate mathematical modeling and transformation processes. It relies on accurate models and requires careful selection of control parameters.

**Integration of Online Model Learning with Model Predictive Control**
A control strategy that bridge the gap between model-based and model-free methods an efficient way to handle the high degrees of freedom and complex dynamics associated with DLOs. Paper [27] propose a framework that integrates online model learning with model predictive control (MPC) for manipulating deformable linear objects (DLOs) like cables and ropes.
This approach requires fewer interaction samples than model-free RL alternatives as it builds a dynamic model of the environment not directly from the data without modeling. It does not rely on pre-collected and it can continually update and refine the model based on ongoing interactions that helps to adapt the changes in object's properties.
Initial training data are collected through random trajectories by executing random actions.
In this control framework, a neural network model is initialized and continually updated based on the collected data. This model predicts the next state of the DLO based on its current state and the action taken. At each time step, MPC is used to optimize the control actions over a finite horizon based on the predictions from the learned model. Only the first action from the optimized sequence is executed, in the next time step the optimization problem is then solved again with the latest state.
The MPC process involves bringing predictions of future states of the DLO us-

ing the learned dynamics model and optimizing control inputs to maximize the cumulative reward over the specified horizon. The reward function is typically based on the difference between the current and desired states, often formulated as the negative Euclidean distance between the achieved and target shapes of the DLO. The optimization is performed using gradient-based methods, using the differentiable nature of the neural network model. This allows the MPC to iteratively adjust the control inputs to steer the DLO towards the desired configuration.
This combination of online model learning and MPC allows efficiently handle complex of DLOs by learning from experience and continually adapting to new data, leading to better performance and adaptability.

This strategy is sample-efficient because it needs fewer interaction samples, and it adapts well by continually updating the model. It is learning directly from interactions and it can handle various tasks. The integration of online model learning with model predictive control (MPC) allows for effective control. However, there are some challenges. Initial data collection requires random actions, which can be inefficient and unsafe (as executing random trajectories to gather initial training data, can cause the movements are not intended) and also might make it slow learning. It's computationally complex due to real-time optimization. It depends on the quality of the learned model to capture dynamics accurately. And MPC's finite horizon might not be enough for long-term planning, the limited number of future steps it considers when optimizing control actions can lead to situations where the controller makes decisions that seem optimal in the short term but are not the best for achieving long-term goals, especially in complex tasks that need planning for a longer period.

**Model-based Reinforcement Learning Approach**

For controlling and planning the manipulation of deformable linear objects (DLOs) a model-based reinforcement learning (RL) framework called PILCO (Probabilistic Inference for Learning Control) is utilized in the paper [5]. predicts the next state of the DLO based on current state and actions taken by the robot.
Actions are defined as the movements (translation and rotation) of the robot's end-effector, which interacts with the DLO at a specific control point (referred to as $p_0$, the first control point).

The change in position from one point to another in the is like $(p_{0,t+1} - p_{0,t})$, indicating the movement from the current position $p_{0,t}$ to a new position $p_{0,t+1}$. The manipulation includes moving this control point through the defined actions, therefore changing the overall configuration of the DLO.

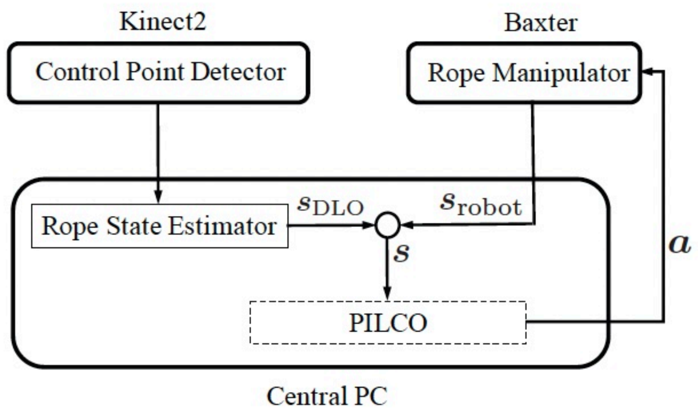


Figure 4.2: The rope manipulation system consists of a Control Point Detector, a Rope Manipulator and a Central PC.

As the robot executes actions and observes the resulting changes in the DLO configuration, these observations feed back into the reinforcement learning model. The model uses this new data to refine its understanding of the DLO dynamics and improve its predictions and its decisions for future actions.

The control strtegy here is very efficient, needing fewer interactions to learn effective control policies. This is important for handling complex tasks like manipulating ropes and cables. It is flexible and does not depend on a fixed model of the object, allowing it to adapt to different objects and environments by learning from observed data. Its model-based approach helps predict future states based on current actions, improving planning and policy quality. The iterative learning process ensures continuous improvement as more data is collected. However, the computational demands of Gaussian Process regression can limit its use with larger datasets and higher-dimensional spaces. The initial phase of random actions can be inefficient or unsafe, which is a concern in sensitive environments. PILCO's reliance on Gaussian assumptions may not always accurately capture complex object dynamics. It has been tested mainly in 2D space, so moving to 3D can have more challenges.

**Differentiable Compliant Position-Based Dynamics**

The control and planning strategy for the manipulation of deformable objects in the paper [14] involves a detailed process of parameter identification and iterative adjustment of control points to achieve a desired rope shape. Initially, the physical parameters of the rope, such as stiffness, are identified through a real-to-sim transfer process. This involves optimizing these parameters by minimizing the difference between the simulated rope and real-world observations, using automatic differentiation techniques. The XPBD framework is used to accurately model the rope's behavior, ensuring the constraints such as shear/stretch, bend/twist, and distance are satisfied.

Once the parameters are identified, the control strategy focuses on achieving a target shape for the rope. This target shape is defined by key points or segments that the rope should conform to. The optimization problem is then formulated with a loss function that measures the discrepancy between the current simulated shape of the rope and the target shape. An iterative control loop is employed, where the XPBD solver simulates the current state of the rope, the loss function is computed, gradients are calculated using automatic differentiation, and control points are updated using gradient descent. This iterative process continues until the rope's shape closely matches the target shape. Experimental validation on the Baxter and dVRK robots demonstrates the effectiveness of this strategy, with the Thomas XPBD solver used for inextensible ropes and the Jacobi XPBD solver for extensible ropes, achieving accurate manipulation and control of the deformable objects.

It utilizing a differentiable framework for effective parameter identification and optimization, which is crucial for fine-tuning simulations to match real-world behavior. It has flexibility and robustness in various applications, including robotic platforms like Baxter and the da Vinci research kit, making it adaptable for different scenarios. However, the strategy involves significant computational complexity due to the need for detailed modeling and iterative solver processes, and implementation complexity requiring a deep understanding of physical modeling and automatic differentiation. Moreover, the accuracy of the model is highly dependent on precise parameter tuning.

**Integrating Control Barrier Function and Quadratic Programming Framework**
In the paper [1] PBD is used to model the object that involves discretizing the objects into particles connected in a mesh or sequence, allowing the prediction of the object's behavior under various forces and movements. The control framework includes a nominal controller for each assistant robot, which aims to minimize the error between its current and target positions relative to a lead robot. This error minimization is critical for maintaining the desired configuration during manipulation tasks.

To ensure safety, the strategy incorporates CBFs that enforce constraints to avoid collisions, prevent overstretching of the object, and maintain safe distances between robots. These functions modify the control inputs dynamically to ensure the system operates within safe parameters. A quadratic programming (QP) based controller integrates the nominal control inputs with the CBF constraints, ensuring that the robot actions remain safe while effectively following the lead robot. The entire approach is implemented in ROS, having real-time control and coordination among multiple robots. The system's architecture supports efficient and safe manipulation of deformable objects in various scenarios, as demonstrated by the successful simulation results presented in the paper.

It uses Position-Based Dynamics (PBD) for real-time motion prediction, helping to avoid obstacles effectively. Control Barrier Functions (CBFs) ensure safety by preventing collisions, overstretching, and keeping agents at safe distances. The Quadratic Programming (QP) framework provides safety with precise control, making it adaptable for different deformable objects, as shown in simulations. Real-time feedback from PBD simulations allows for quick adjustments. However, there are some downsides. The method is computationally heavy. The testing has been in simulations, not real-world settings. It assumes obstacles are static, which is a limitation.

**A framework including state estimation, task planning and trajectory planning**
Complex tasks such as knotting ropes in manipulation of deformable objects is addressed by in the paper[23].
The core of this framework relies on the Coherent Point Drift (CPD) algorithm, which aligns the object's estimated positions from the previous time step with

current point cloud measurements from 3D cameras. This method allows for robust real-time state estimation, even in the presence of noise and occlusions. By discretizing the DLO into a set of nodes and using CPD to register these nodes with the point cloud data, the system can effectively track the object's state.

Task planning is achieved by comparing the current state of the DLO with a set of pre-recorded states using CPD. The system calculates a log-likelihood value for each comparison to measure similarity and identify the current step in the manipulation process. If the similarity is below a certain threshold, indicating a failure, the system can prompt human intervention to correct the state, which is then added to the set of known states for future recovery. For trajectory planning, the framework uses a learning-from-demonstration approach, where human operators demonstrate manipulation trajectories for specific initial states. During execution, these trajectories are adapted to the current state using the transformation function generated by CPD, ensuring accurate and feasible manipulation paths for the robot's end-effectors. This closed-loop control strategy allows for efficient and robust manipulation of DLOs, addressing challenges like variability in object shapes and environmental conditions.

This control strategy has robust state estimation, effectively handling occlusions and noisy sensor data, and integrates state estimation, task planning, and trajectory planning into a coherent framework, simplifying implementation and increasing efficiency. It adapts to real-time changes, and make sure flexible manipulation, and benefits from human demonstrations, ensuring intuitive and effective robot movements, while including mechanisms for failure detection and recovery to improve reliability. However, it faces challenges such as computational complexity due to the intensive nature of CPD, dependence on sensor data quality for accurate state estimation, the need for initial manual demonstrations which can be time-consuming, potential difficulties in handling extremely complex deformations which might reduce accuracy, and limited generalization to new types of deformable objects, requiring additional training for different objects.

**Kinematic Chain**

Kinematic chain model that is used for modeling, simplifies the complex dynamic of deformable object into manageable model for control in the paper [29].

A camera as an observer to generate point cloud data of the DLO in real-time is used. This data provides a direct observation of the DLO's shape and position. With the point cloud data, the observer estimates the state of the DLO by aligning the observed shape with the simulated model. This process involves calculating the deviation between the actual DLO and its simulated one and adjusting the simulation to reduce this deviation. The observer calculates corrective forces based on the deviations detected in the point cloud, updating the multibody simulation to align it more closely with the observed state of the DLO.
The controller operates based on the kinematic model of the DLO, utilizing the Jacobian matrix that relates joint velocities to changes in the position and orientation of the DLO.
The controller is tasked with following a predefined trajectory. It calculates the required joint velocities to minimize the positional error between the current state of the DLO (as updated by the observer) and the desired trajectory.
Then it uses a feedback loop where the positional error influences the velocity inputs, ensuring that the controller adapts to any deviations observed by the point cloud data.
It does not need prior knowledge about the DLO's physical properties, making it easier to use with different DLOs. It is highly accurate, and uses point cloud data effectively to keep track of the DLO's state. The system also allows for different processing speeds for the camera and simulation, making it more flexible. However, there are some drawbacks. The system requires a lot of computational power. It is complex to implement because of the need for detailed simulations and point cloud processing.

**Dynamic Control Schemes for Single-Arm and Dual-Arm**

The paper [16] presents control and planning strategies for manipulating deformable linear objects (DLOs) using robotic arms, focusing on both single-arm and dual-arm control schemes. In the single-arm control strategy, one end of the DLO is fixed, and the other end is controlled by a robotic arm to conform to a predefined target curve on a plane. The target curve is divided into discrete points, and the arm's movement is adjusted dynamically based on the errors between the current position of the DLO points and the target points. This dynamic adjustment ensures the DLO progressively matches the target curve as the arm moves, with control parameters fine-tuned through simulations and validated in real robotic experiments.

The dual-arm control strategy involves manipulating both ends of the DLO with two robotic arms. Initially, the DLO is positioned above the target plane, and the arms move vertically in sync to bring the DLO into contact with the plane. Once a starting point on the DLO aligns with the corresponding point on the target curve, the arms move to shape the DLO according to the target curve. The strategy ensures synchronized movements of both arms, preventing any undesired stretching or distortion of the DLO. Errors between the DLO points and the target points are calculated, and the horizontal velocities of the arms are adjusted dynamically. A nontime perceptive reference-based coordination method ensures both arms maintain synchronized movement, even if one arm's movement is interrupted.

Both strategies emphasize preprocessing the target curve, calculating positional errors, and dynamically adjusting the robot arm velocities based on these errors. Control points for the arm movements are generated through simulations and validated through real robotic experiments. The single-arm control focuses on dynamic adjustments of one end, while the dual-arm control ensures coordinated manipulation of both ends for precise and synchronized shaping of the DLO.

This includes a detailed model that accurately captures stretching, bending, and twisting of the DLO, effective single-arm control for simple tasks by adjusting the robot's motion based on errors, and precise dual-arm control for complex shapes through coordinated movements. However, there are challenges such as the difficulty and higher computational demands of coordinating two robotic arms, the limited ability of single-arm control for complex shapes since it only moves one end.

**Real-time Physically-accurate Simulator for Long Flexible Cable Manipulation**

The paper [12] focuses on developing a simulation framework that supports the control and planning of manipulating deformable objects, specifically long flexible cables. The proposed method discretizes the cable into multiple rigid cylindrical-link segments, each modeled with a complementarity-based contact model and compliant coupling based on Cosserat rod theory. To enhance computational efficiency, the cable is divided into subsystems of consecutive

segments, allowing parallel solving of each subsystem's dynamics. This parallelized subsystem approach, combined with passive midpoint integration (PMI), ensures stable and accurate simulation.
The framework includes a novel Gauss-Seidel based iterative solver that maintains inter-subsystem consistency through iterative enforcement of coupling and contact impulses. To further improve convergence and simulation speed, a projected Gauss-Seidel (PGS) post-regulation scheme is implemented. The simulation accurately handles collision detection and self-collisions using an adaptive subsystem division scheme and a generalized contact model that better represents interactions for materials like polymers.

By integrating manipulator dynamics as additional subsystems, the framework facilitates coordinated control of both the manipulator and the cable. The simulation's real-time performance is validated through experiments, ensuring its accuracy and reliability.
The system is fine-tuned in real-time to match experimental data and ensure the simulation remains accurate and stable. The simulation results are continuously compared against experimental data to ensure they are realistic. Adjustments are made to the solver's parameters to improve convergence and stability, ensuring the simulation runs efficiently without sacrificing accuracy.

**Multivariate Dynamic Splines**

Another approach of control strategy for manipulation of deformable linear objects (DLOs) such as cables and wires is utilizing advanced technology in the form of multivariate dynamic splines that is represented in the paper [21]. The model represents the DLO as a series of control points connected by spline functions. This approach employs the flexibility and adaptability of spline-based models to accurately predict and control the shape changes of DLOs during robotic manipulation. The dynamic splines are utilized to do the DLO's configuration in real-time, enabling precise adjustments in response to dynamic manipulation scenarios.
After constructing a dynamic model of the DLO using multivariate dynamic splines, a manipulation primitive is defined as a specific action performed by the robot on the DLO, such as moving one end to a new location. This step involves defining how these primitives affect the control points of the DLO model. Time-based spline interpolation is used to model how the control points evolve during the execution of a manipulation primitive. This approach simplifies the

continuous dynamic model into a discrete set of control points that can be efficiently computed.
Before manipulation begins, the initial configuration of the DLO is detected and estimated, typically using vision systems. This information initializes the control points for the dynamic model.
During manipulation, the model iteratively updates based on the robot's actions and the resulting DLO behavior. This simulation involves recalculating the positions of control points as the DLO moves, based on the dynamic spline equations. The system uses a feedback loop to compare the predicted shape of the DLO with its actual shape (as observed by sensors). Adjustments are made to the robot's actions based on this feedback to correct any deviations from the desired trajectory.
The trajectory of the manipulation primitives is optimized in real-time to achieve the desired end shape of the DLO. This involves minimizing a cost function that can include factors like the energy used, time taken, and accuracy of the DLO configuration. Using an iterative approach, the control strategy adjusts the manipulation primitives to handle dynamic changes in the DLO's behavior and external disturbances.

It has real-time prediction and control due to dynamic splines enabling precise adjustments during manipulation, flexibility and adaptability through spline-based models that works for various shapes and configurations, integration of a feedback loop that allows continuous error correction by comparing predicted and actual shapes, and computational efficiency by reducing the dynamic model to discrete control points. However, it also has drawbacks such as dependency on approximate solutions (simplified model) that may not capture all physical behaviors, sensitivity to parameter selection requiring fine-tuning, dependency on vision systems for initial configuration detection that can be error-prone, potential convergence issues with the iterative algorithm especially in complex tasks, and the need for robust computational hardware to meet real-time requirements.

**Global Planning and Local Control**

The paper [19] presents a comprehensive strategy for dual-arm manipulation of deformable linear objects (DLOs) by combining a global planning approach with local control. The global planner uses a Constrained Bi-directional Rapidly-

Exploring Random Tree (CBiRRT) framework, which efficiently explores the configuration space and finds a collision-free path from start to goal configurations. This method uses a simple mass-spring DLO energy model to project random configurations onto stable ones, ensuring the feasibility of the path. The planner grows two trees, one from the start and one from the goal, and iteratively extends them towards each other until they connect, forming a coarse path that avoids obstacles.

The local control strategy employs Model Predictive Control (MPC) to refine the execution of the coarse path, compensating for modeling errors and ensuring precise manipulation. The controller uses real-time feedback to adjust the robot's movements, employing artificial potentials to drive the DLO and robot arms towards desired configurations while avoiding obstacles. The DLO's configuration is continuously updated using a learned Jacobian matrix, and the control input is optimized to minimize deviations from the planned path. This integrated approach ensures robust and accurate manipulation of DLOs in constrained environments, effectively combining the strengths of global planning and local control.
The control strategy here has high precision in manipulation because it uses both global planning and local control. It effectively avoids obstacles through both planning and real-time adjustments. The strategy is robust to modeling errors thanks to real-time feedback, and it is flexible enough to handle different tasks and environments. The global planning phase is efficient because it uses a simple mass-spring model. Combining global planning and local control ensures strong overall performance. However, The local control phase can be computationally demanding because it uses model predictive control (MPC). It relies on accurate real-time feedback, which can be a problem if the data is incorrect. Tuning the necessary parameters can be time-consuming.

**Learning-Based Model Predictive Control (MPC)**
In manipulation of linear objects in environment with obstacles, a control strategy approach can be a effectively utilize of the predictive power of a learning-based MPC with the robust safety that it is provided by a CBF-based filter. This approach allows for precise and safe manipulation of deformable objects in cluttered or unpredictable environments that is introduced in the paper [24].
The control and planning strategy for manipulating deformable linear objects (DLOs) in this paper combines learning-based Model Predictive Control (MPC)

with a Control Barrier Function (CBF) safety filter. The MPC component utilizes a global deformation model (GDM) that incorporates a self-attention mechanism within a Transformer encoder to accurately predict the DLO's deformation behavior. This model is trained offline with data from unconstrained environments and fine-tuned online. The MPC formulates an optimization problem over a receding horizon, generating reference actions that minimize deviation from the desired configuration, avoid obstacles, and penalize large end-effector velocities. These reference actions guide the DLO towards the target configuration while considering the constraints of the environment.

To ensure safety and address potential inaccuracies in the global model predictions, the CBF safety filter refines the MPC-generated actions. It uses a local linear deformation Jacobian model, updated online, to represent the DLO's local behavior more accurately. The safety filter solves a quadratic programming problem to find the safest feasible actions close to the reference actions, ensuring obstacle avoidance and adherence to control input constraints. This integrated approach combines the predictive capabilities of learning-based MPC with the safety assurances of the CBF filter, enabling robust and effective manipulation of DLOs in complex, constrained environments.
It has good prediction accuracy because it uses a detailed model of how the DLO changes with control inputs. The CBF-based filter adds a strong safety to avoid collisions. This approach is adaptable to different DLO setups and has been proven effective through many tests and real-world experiments. However, there are some challenges. The method can be computationally heavy because the MPC and safety filter need a lot of calculations. It relies on the accuracy of the learned model, which might not work well if the training data isn't good enough. The need for continuous model updates adds extra computational load, which can be challenging to manage in real-time.

### 4.1.2 Model-Free Control

**Deep Deterministic Policy Gradient (DDPG)**
The main focus of a research that is done in paper [11] is using reinforcement learning (RL) methods.
The problem is complex because it involves controlling not just the position but also the shape of the object, which can change in non-linear ways due to its

elastoplastic properties.
Deep Deterministic Policy Gradient (DDPG) method that is a model-free reinforcement learning approach is chosen to allows the system to learn effective control strategies directly from interactions with the environment, without needing a physical model of the DLO.
The DDPG learns a policy that maps the observed state of the DLO and the gripper to actions that aim to achieve the desired shape.
The state space that includes the physical and geometric properties of the object, such as its curvature and torsion, as well as the state of the robotic gripper handling it, which includes positions and velocities. The action space is defined by the velocity commands to the Cartesian gripper, that directly affect the manipulation of the DLO. The actor network in DDPG learns a deterministic policy mapping these states to the actions that aim to achieve a desired shape of the DLO, optimized by continuous adjustment of the network parameters.

The reward function is necessary in this approach as it guides the learning process. It is designed to penalize the difference between the current shape of the DLO and the desired shape. The function can include terms for the accuracy of the shape achieved. There is actor and critic. The actor network proposes actions (manipulations by the gripper) based on the current state, and the critic network evaluates these actions by estimating the reward outcomes. Through training, the actor's policy is continuously adjusted to maximize the expected rewards.
This method is flexible because it doesn't need a physical model of the object, making it suitable for various types of DLOs. It can handle the complex behaviors of materials that change shape both elastically (reversible) and plastically (permanent). The method uses a detailed state space that includes the shape and motion properties of the object, and the continuous action space allows for precise control. The reward function is designed to help the system learn by penalizing deviations from the desired shape. However, there are some drawbacks. The method requires a lot of data to learn effectively, which can be time-consuming and expensive. The method also needs significant computational resources for training and relies on accurate state estimation.

## 4.2 Planar

Manipulation of highly deformable materials like cloth is a very challenging problem due to their high dimensionality and also needs for multiple grasp points to fully manipulate the material.

### 4.2.1 Model-Based Control

**Task-level Optimization and Feedback Control**
For manipulation of cloth-like materials that is a complex task that paper [10] shows an approach with human-robot interaction can effectively used to handle tasks. The collaborative approach ensures that the manipulation is performed efficiently and safely, with the human providing oversight and the robot handling precise, repetitive movements.
After system description and modeling the object, the next step of the manipulation is task-level optimization.
It involves determining optimal position and velocity setpoints for the robot based on human pose information. With using the position of the human, the system computes an optimal pose for the robot that minimizes the internal force of the cloth, promoting efficient and damage-free handling.
In the next step, a feedback control law that adjusts the robot's motion in real time to align with the human's changes in position is used. This real-time adjustment is crucial for handling the unpredictable dynamics of cloth.
A quadratic program is formulated to solve for control signals that ensure the robot achieves the desired end-effector velocities, adhering to both dynamic and kinematic constraints.

It makes handling efficient by combining human decision-making with robot precision, it adjusts in real-time to changes in the material using a feedback control law, it finds the best robot positions to avoid damaging the material, and it follows rules to keep the robot's movements safe and effective using a quadratic program. However, there are some drawbacks. It depends a lot on human input, which limits robot independence, it's complex and requires a lot of computing power due to sophisticated modeling and optimization, the robot's base can't handle tight turns well because of nonholonomic constraints, and it can fail with complicated movements.

**Model-Based Motion Planning Control**
Dynamic Fabric Simulator (DFS), a particle-based model that is used in the Paper [18] simulates the real-time behavior of deformable materials through interconnected mass-spring systems. Complementing DFS, the Multi-Agent Handling Planner (MAPL) uses inverse planning techniques to orchestrate the movements of robotic agents based on the actions observed from human operators via the Operator Tracking for Co-manipulation (OTC) module.
The outputs from the DFS and OTC into the MAPL, which then calculates how robots should move to assist the human operator. This includes determining optimal grasping points on the fabric and adjusting robot positions to prevent fabric damage or inefficiency in handling.
Parameters of the model (like spring stiffness and damping in DFS) can be dynamically adjusted based on real-time feedback and empirical data, allowing the system to adapt to different fabrics and manipulation requirements.

It can simulate fabric behavior in real-time, plans robot movements based on human actions, adapts to different fabrics by adjusting settings, and prevents fabric damage. It helps humans and robots work well together in factories by tracking human actions with the Operator Tracking for Co-manipulation module. However, it also has some downsides. Setting it up is complex because it involves integrating different parts, it needs a lot of computing power for real-time performance.

**Model-Based Iterative Control Strategy**
In the paper [7] the control strategy for manipulating deformable objects involves a methodical process of simulation and iterative refinement. The procedure starts by placing the target object on a working platform and aligning it with the corresponding simulation model. Initial configuration properties, such as key interest points, are determined to ensure a consistent relationship between the simulated model and the real-world object. Velocity is then applied to the model's particles at positions corresponding to where the robotic manipulator will grasp the object, simulating the expected deformation and manipulation outcome. This process is iterative, involving continuous adjustments to the grasping points and re-simulations until the handling process is successful in the simulation. Once a satisfactory manipulation strategy is identified in the

simulation, it is applied in real-world experiments to observe and verify the handling task. These real-world experiments help in validating the simulation results, ensuring the model's predictions are accurate and the robotic manipulation system can effectively handle flexible deformable objects. This strategy ensures that the model is robust and effective for tasks such as hanging, spreading, and folding.
It has versatility, accurate prediction, and robust calibration, allowing for fine-tuned planning and execution. However, the approach has challenges such as computational complexity, reliance on approximations, sensitivity to initial conditions, and limited handling of multiple manipulators and collision detection.

**Adaptive Control and Planning, Using Latent Representation and Graph Dynamics**
The control strategy in the paper [15] is using a latent representation of the object's physical properties. This representation is extracted by an adaptation module from exploratory actions, such as a pulling interaction. The adaptation module processes these observations to generate a latent representation that encapsulates the elastic properties of the object. This representation is then utilized by a forward dynamics module, which predicts the future states of the deformable object based on this latent information. The forward dynamics module employs a Graph Neural Network (GNN) to model the object's state as a graph, enabling accurate predictions of how the object will respond to various control actions.

For control and planning, there is a training an inverse dynamics model, which calculates the control actions needed to achieve specific state transitions. By providing an initial and a desired state of the deformable object, this model determines the necessary actions to manipulate the object effectively. Moreover, the learned latent representations can be transferred to different tasks and environments, enhancing the model's generalizability. This allows for the application of the model to various manipulation tasks, such as lifting or bandaging, both in simulation and real-world scenarios. The approach shows robust adaptability, as the models are able to generalize to new deformable objects with different physical properties, enabling effective and flexible manipulation strategies in robotic systems.

**G-DOOM (Graph dynamics for DefOrmable Object Manipulation)**

Another approach for manipulation of deformable object is G-DOOM (Graph dynamics for DefOrmable Object Manipulation) that performs visual manipulation that is introduced in the paper [17].
After graph-based dynamics modeling of these objects that is keypoints detection and graph construction, the G-DOOM framework incorporates a recurrent neural network (RNN) with the GNN. As it's important to consider not only where the keypoints are located on the object but also how they move and interact as the object is manipulated over a period of time and this combination tracks the changes of keypoint interactions over time, thus maintaining a state about the object's configuration across frames.

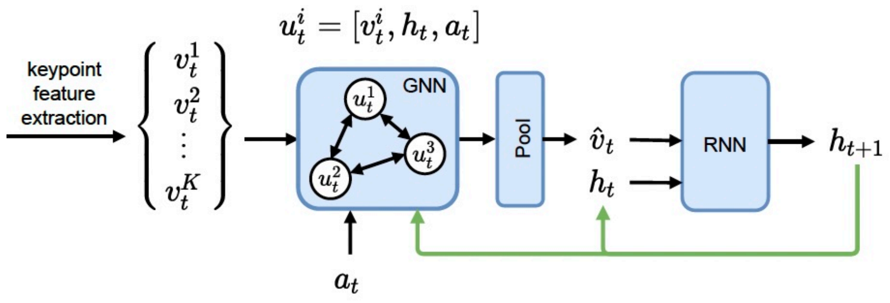


Figure 4.3: compose keypoint features into a graph, learning recurrent graph dynamics

The next step is applying model-predictive control (MPC) to predict and plan the necessary robot actions to manipulate the object towards a desired state. This includes computing the future states of the object under different possible actions and selecting the action such as folding a cloth.

And there is a real-time adjustment that as actions are executed, new visual data are collected and processed to update the keypoints and their dynamics. The MPC loop uses this updated information to refine future actions continually.

The G-DOOM control strategy has many strengths. It uses a graph of keypoints with graph neural networks (GNN) and recurrent neural networks (RNN) to effectively model the complex movements of deformable objects. This method makes it easier to understand and predict how objects change shape over time, even when parts of the object are hidden. Model-predictive control (MPC) helps plan and adjust actions in real-time, making the system robust and adaptable in both simulated and real-world tests. However, there are some downsides. Building and processing these graphs takes a lot of computing power, and relying on a few keypoints might miss some details. Training the system requires a lot of data.

### 4.2.2 Model-Free Control

Model-free control strategy instead of relying on a predefined physical model of the material (model-based), this approach uses learned behaviors from data. The CNN acts as a black-box predictor that estimates deformations based on visual inputs, which the control system then uses to adjust the robot's actions.

**Convolutional Neural Networks (CNNs)**
Paper[20] involves data-driven approach using an ensemble of Convolutional Neural Networks (CNNs) and a 3D camera for depth mapping.

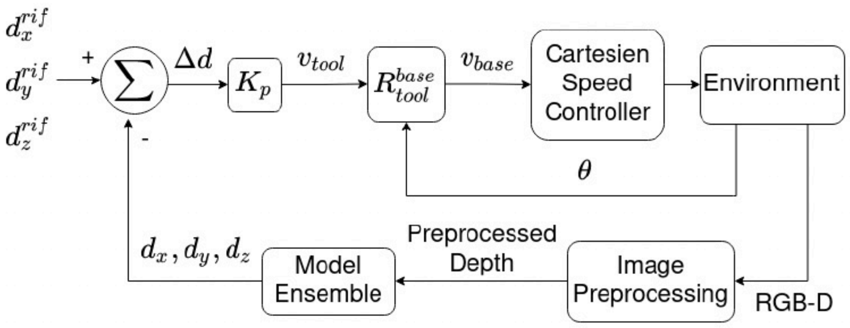


Figure 4.4: Control scheme.

The first step for manipulation of soft material in this paper is data collection that it collects depth images of the deformable material (carbon fiber ply) under various configurations using a 3D camera. These images represent different states of deformation caused by relative human-robot displacement.
In the preprocessing step these images are preprocessed to segment the material and filter out unnecessary background information, enhancing the accuracy of deformation detection.
In neural network training step multiple CNNs are trained on the preprocessed

images to predict the deformation of the material based on the visual data. The network outputs an estimated displacement of the material relative to its nominal position.
In control implementation part, the estimated displacement information is used to calculate a control command for the robot. A proportional controller converts the displacement error (difference between current and desired positions) into velocity commands for the robot's tool.

The control strategy in the paper has several benefits, including efficiency and adaptability because it uses a data-driven approach that avoids complex simulations and learns from visual data. The method is accurate and robust. It's easy to implement with a simple proportional controller that converts displacement errors into velocity commands (as the proportional controller takes the estimated displacement error and it is straight forward method), and CNNs automatically extract useful visual features. However, there are some drawbacks. It doesn't handle rotational movements. It can be sensitive to noise and inaccuracies at high speeds due to averaging depth images, which can blur the input. (there is trade-off between noise reduction and accuracy at higher speed).

## 4.3 Three-Dimensional

The objects with three dimensional shape, they can include anything from soft materials to complex structures. Managing and controlling these deformable objects involves understanding their physical properties and developing strategies to manipulate them effectively.

### 4.3.1 Model-Based Control

**FEM-based Deformation Control for Dexterous Manipulation of 3D Soft Objects**
In the paper [3] after modeling the deformable object using the Finite Element Method (FEM), the control strategy for manipulation focuses on computing the necessary contact forces to achieve a desired deformation. This is framed as an inverse simulation problem where Lagrange multipliers represent the force intensities applied at the contact points by the robotic fingertips. The optimiza-

tion problem aims to minimize the displacement difference between actual and desired positions of end-effector points, projected into the constraint space using the Schur complement for computational feasibility. This process involves a multi-rate control loop, with a low-frequency loop for FEM computations and a high-frequency loop for solving the Quadratic Programming (QP) problem and controlling the actuators in real-time.

For the dexterous manipulation of the object, the SCHUNK 5-Finger Hand (S5FH) is used, with its kinematics modeled to compute joint displacements required for desired fingertip positions. The closed-loop inverse kinematic algorithm (CLIK) controls the hand, addressing errors due to underactuation and sensor inaccuracies. The desired deformation can be specified via a GUI or predefined trajectories. While the initial approach is open-loop, estimating elasticity parameters beforehand helps minimize errors. Future improvements aim to incorporate vision feedback to enhance accuracy and reduce errors from model approximations and mechanical limitations.

The initial control strategy is open-loop, which means there is no feedback mechanism to correct errors in real-time during the manipulation process. This can lead to inaccuracies.

**Using Real-Time Visual Tracking and Elasticity Parameter Estimation**
The control strategy in the paper [22] centers on real-time deformation tracking using an RGB-D camera that captures the object's deformation. The initial pose of the object relative to the camera is determined using pre-trained markers. The visual tracking data drives a physics-based simulation to update the object's state, minimizing the point-to-plane error between observed deformations and the simulated model. This iterative process employs a Jacobian matrix and an Iteratively Reweighted Least Squares (IRLS) scheme to continually refine the tracking accuracy.

The planning strategy involves an adaptive process where the robot manipulates the object while continuously capturing its deformation with the RGB-D camera. The visual data updates the FEM simulation in real-time, reflecting the actual deformations. During manipulation, the system estimates the elasticity parameters, such as the Young's modulus, using measured external forces applied to the object by the robot's end-effector. This estimation is achieved through the Levenberg-Marquardt optimization, which iteratively adjusts the parameters to minimize the error between tracked and simulated deformations.

Once the elasticity parameters are accurately estimated, the system can perform remote force estimation using visual data alone, eliminating the need for a force sensor. This enables the robot to predict and apply precise forces for desired deformations, enhancing control strategies. The method handles occlusions and errors robustly.

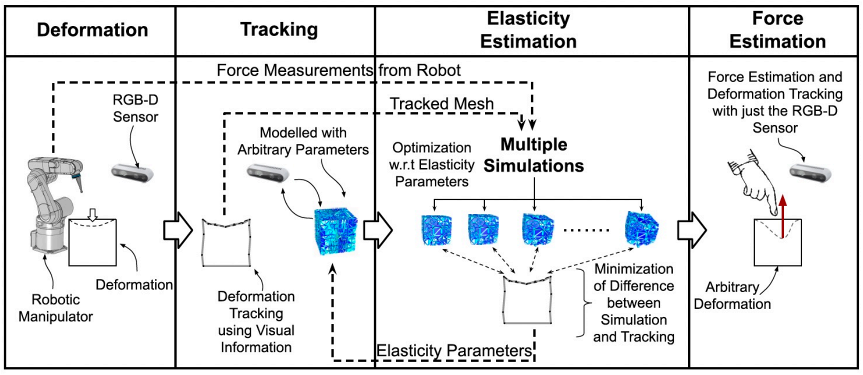

Deformation
Tracking
Elasticity Estimation
Force Estimation
Force Measurements from Robot
RGB-D Sensor
Tracked Mesh
Modelled with Arbitrary Parameters
Optimization w.r.t Elasticity Parameters
Multiple Simulations
Force Estimation and Deformation Tracking with just the RGB-D Sensor
Deformation
Robotic Manipulator
Deformation Tracking using Visual Information
Minimization of Difference between Simulation and Tracking
Elasticity Parameters
Arbitrary Deformation

Figure 4.5: The STEPE and Remote Force Estimation architecture.

The control strategy here has real-time adaptability, it works with various soft objects without needing to know their properties beforehand, combines visual and force data for accurate tracking, and improves precision by continuously updating its model. It can also predict forces without direct measurements and works with different shapes and complexities of objects. However, there are some downsides. The method requires a lot of computational power, relies on accurate sensors, can be affected by lighting and occlusions, only works well for small deformations.

### 4.3.2 Model-Free Control

**Robust Shape Estimation for Deformable Object Control**

In the paper [6] the control and planning strategy for manipulating deformable objects is centered around a robust, real-time shape estimation approach. This method uses an RGB-D camera to capture depth images of the object and employs a joint tracking and reconstruction framework to continuously update and refine the object's shape model. The tracking component estimates the deformation model by aligning a reference shape model with the live input from the camera, while the reconstruction component integrates multiple RGB-D frames into the reference shape model to enhance its accuracy and detail.

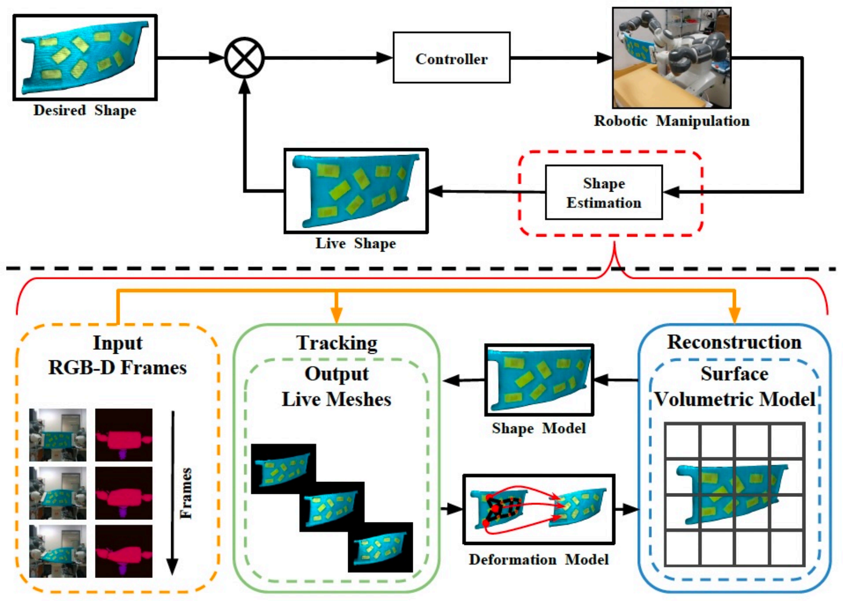


Figure 4.6: Shape Estimation Pipeline

The system operates in a shape control framework, where the robot iteratively estimates the current shape of the deformable object and uses the difference between the current and desired shapes to generate control signals. The deformation model, crucial for this process, combines a global rigid transformation and local non-rigid deformations represented by a sparse deformation graph. This model allows the system to handle complex deformations in real-time, ensuring that the shape estimation remains accurate even under challenging conditions like noise and occlusions.

The control strategy here includes real-time performance due to efficient parallel processing, model-free flexibility allowing adaptability to various deformable objects without prior knowledge, robustness to noise by integrating multiple frames, and robustness to occlusion through joint tracking and reconstruction. Additionally, it provides high-quality shape reconstruction by maintaining a continuously updated high-resolution reference model. However, it has some limitations, such as dependency on precise deformation estimation, challenges in handling large deformations and complex topological changes due to limitations of the ARAP regularization term, It requires significant computational power, especially for the GPU-based solver. Lastly, errors can accumulate over time, reducing accuracy.

**Deep Neural Networks (DNNs)**
An approach that is learning-based, model-free is presented in paper [8] , utilizing deep neural networks (DNNs), the strategy avoids the need for pre-defined physical models, learning directly from sensory data captured during manipulation tasks. And it is not only robust but also adaptive, continuously refining its performance through an online learning process.

The core of the control strategy involves a 5-layer deep neural network (DNN) designed to approximate the deformation function. This network maps the feature velocities (changes in the feature vector) to control velocities for the robotic end-effector. The DNN is trained online, meaning it continually updates its parameters using real-time data collected during the manipulation process. This online learning approach ensures the model becomes increasingly accurate and robust over time.

The control process is executed by computing the difference between the current feature vector and a target feature vector, which represents the desired state of the object. The DNN uses this difference to predict the necessary control velocities to minimize the gap, guiding the robotic end-effector to achieve the desired deformation. Additionally, an occlusion recovery algorithm is employed to maintain a complete point cloud representation, even when parts of the object are occluded during manipulation. This involves real-time tracking and reconstruction to fill in any missing data.

The output of the deep neural network (DNN) is the control velocity for the robotic end-effector. This control velocity is a six-dimensional vector representing the required movements along the x, y, and z axes, to manipulate the deformable object towards its desired state.

The DNN takes the feature velocities (which are the changes in the feature vector derived from the 3-D point cloud data) as input. It then processes this input through its layers to predict the control velocities that will minimize the difference between the current state of the deformable object and the target state. These predicted control velocities guide the robotic end-effector's movements to

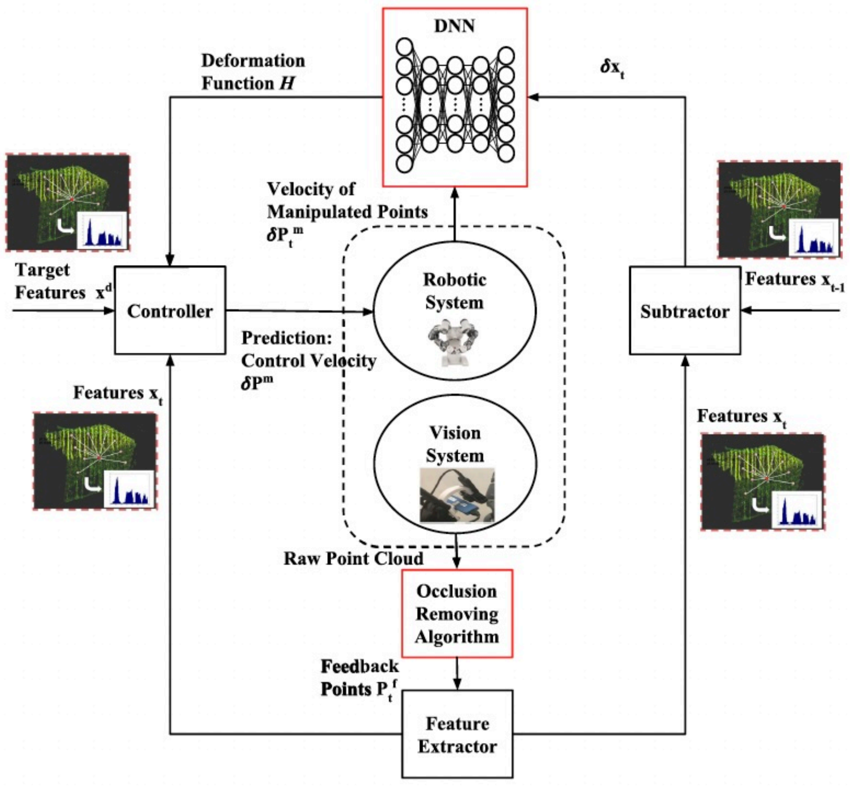


Figure 4.7: The Online Learning Process

achieve the desired deformation of the object.

This control strategy doesn't need predefined physical models, making it flexible for different deformable objects by learning directly from sensory data. Using deep neural networks gives it strong representation power and the ability to improve continuously through online learning, adapting to new data in real-time. However, there are some downsides. It requires a lot of computational power due to the complexity of the neural networks and real-time processing. It depends on accurate real-time tracking and reconstruction to handle occlusions, which can affect its performance if not done correctly.

# 5
# Conclusion

This thesis offers a comprehensive analysis of contemporary modeling and control strategies for the manipulation of deformable objects. The research categorizes existing methodologies into three approaches: analytical, numerical, and data-driven for modeling; and model-based, model-free control strategy. We evaluates the strengths, limitations, and suitable applications of each technique, providing separate assessments for both modeling methods and control strategies.

Initially, we proceed through the modeling section:
Analytical methods for linear deformable objects (DLOs), has high precision and theoretical clarity, making them suitable for well-defined systems. They excel in applications where the physical properties and boundary conditions are accurately known and do not change significantly. However, they are limited by their assumptions and may not capture the full complexity of real-world behaviors. Numerical methods handle complex geometries and material properties, providing detailed simulations of DLO behavior under various conditions. Moreover, they are adaptable to different scenarios but can be computationally intensive and require careful parameter tuning. Data-driven methods excel in handling nonlinearities and complex interactions that are challenging for traditional modeling techniques. They are particularly useful when the underlying physics is not well understood or when the material properties are highly variable. However, they require significant computational resources and high-quality training data. Applications include real-time control in dynamic environments and adaptive manipulation tasks.

Numerical methods for planar deformable objects, for materials like cloth and fabrics have detailed and accurate simulations by approximating the object's behavior under various forces. However, they are computationally demanding. Data-driven approaches for planar objects capture complex interactions within fabrics. These models are adaptable and generalizable, making them suitable for real-time applications where speed and flexibility are crucial. However, they depend heavily on the quality and quantity of training data.
Analytical methods for 3D deformable objects includes representing the object using mathematical equations based on physical laws. These methods, though less commonly used than numerical methods, they can be useful in specific, simpler scenarios. Numerical methods for 3D deformable objects has detailed presentation of complex behaviors like bending and stretching. These methods ensure physical accuracy but they have high computational complexity and require careful parameter tuning.

We next proceed to the control strategy section, which constitutes the subsequent necessary step for the manipulation of deformable objects:
Model-based control strategies for DLOs focus on reducing oscillations at the DLO's end, ensuring stability and precision. These strategies require accurate modeling and are effective in achieving smooth operations but can be complex to implement due to the need for detailed mathematical models and parameter tuning. Model-free control approaches have flexibility and adaptability without needing a predefined physical model. These methods excel in handling various DLOs and their complex behaviors but require extensive data for training and good computational power for real-time application.
The control strategies for planar deformable objects, such as fabrics and sheets, present unique challenges due to their high dimensionality and the need for multiple grasp points. Model-based approaches has precise control and real-time adjustments, making them suitable for applications requiring detailed manipulation of planar objects. These methods are effective in scenarios where accurate modeling of the object's behavior is needed and computational resources are adequate to support complex simulations. Model-free strategies, uses data-driven techniques provide robust and adaptable solutions that can learn from experience. These methods are particularly advantageous in environments where the object's properties are difficult to model accurately. However, they require major

amounts of training data and are sensitive to noise and inaccuracies, particularly at high speeds.
The model-based for 3D provide high precision by utilizing detailed physical models. These methods are suitable for applications requiring accurate control of the object's shape and deformation, it is ensuring precise manipulation. However, these strategies are computationally demanding. Model-free strategies, has significant flexibility by learning from data. These approaches are advantageous in environments with unpredictable dynamics, as they do not rely on predefined physical models. However, they necessitate extensive training data and computational power, and can be sensitive to noise and inaccuracies, especially at high speeds.

# References

[1] Burak Aksoy and John Wen. *Collaborative Manipulation of Deformable Objects with Predictive Obstacle Avoidance*. 2024. arXiv: `2401.16560 [cs.RO]`.

[2] Feng Ding et al. “Anti-swing Control in Manipulation of a Deformable Linear Object using Dynamic Surface Control”. In: *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*. 2018, pp. 503–508. DOI: `10.1109/ICARM.2018.8610788`.

[3] F. Ficuciello et al. “FEM-Based Deformation Control for Dexterous Manipulation of 3D Soft Objects”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 4007–4013. DOI: `10.1109/IROS.2018.8593512`.

[4] N. Roca Filella et al. “3D visual-based tension control in strip-like deformable objects using a catenary model”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 3210–3217. DOI: `10.1109/IROS47612.2022.9982197`.

[5] Haifeng Han, Gavin Paul, and Takamitsu Matsubara. “Model-based reinforcement learning approach for deformable linear object manipulation”. In: *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. 2017, pp. 750–755. DOI: `10.1109/COASE.2017.8256194`.

[6] Tao Han et al. *Robust Shape Estimation for 3D Deformable Object Manipulation*. 2018. arXiv: `1809.09802 [cs.RO]`.

[7] Yew Cheong Hou, Khairul Salleh Mohamed Sahari, and Dickson Neoh Tze How. “Modeling of flexible deformable object for robotic manipulation”. In: *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*. 2016, pp. 92–95. DOI: `10.1109/IRIS.2016.8066072`.

[8] Zhe Hu et al. "3-D Deformable Object Manipulation Using Deep Neural Networks". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4255–4261. DOI: 10.1109/LRA.2019.2930476.

[9] Chunli Jiang et al. "Dynamic Flex-and-Flip Manipulation of Deformable Linear Objects". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 3158–3163. DOI: 10.1109/IROS40897.2019.8968161.

[10] Daniel Kruse, Richard J. Radke, and John T. Wen. "Human-robot collaborative handling of highly deformable materials". In: *2017 American Control Conference (ACC)*. 2017, pp. 1511–1516. DOI: 10.23919/ACC.2017.7963167.

[11] Rita Laezza and Yiannis Karayiannidis. "Learning Shape Control of Elastoplastic Deformable Linear Objects". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 4438–4444. DOI: 10.1109/ICRA48506.2021.9561984.

[12] Jeongmin Lee et al. "A Parallelized Iterative Algorithm for Real-Time Simulation of Long Flexible Cable Manipulation". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 12040–12046. DOI: 10.1109/ICRA48506.2021.9561905.

[13] Xiang Li, Zerui Wang, and Yun-Hui Liu. "Sequential Robotic Manipulation for Active Shape Control of Deformable Linear Objects". In: *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. 2019, pp. 840–845. DOI: 10.1109/RCAR47638.2019.9044123.

[14] Fei Liu et al. "Robotic Manipulation of Deformable Rope-Like Objects Using Differentiable Compliant Position-Based Dynamics". In: *IEEE Robotics and Automation Letters* 8.7 (2023), pp. 3964–3971. DOI: 10.1109/LRA.2023.3264766.

[15] Alberta Longhini et al. "EDO-Net: Learning Elastic Properties of Deformable Objects from Graph Dynamics". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 3875–3881. DOI: 10.1109/ICRA48891.2023.10161234.

[16] Naijing Lv, Jianhua Liu, and Yunyi Jia. "Dynamic Modeling and Control of Deformable Linear Objects for Single-Arm and Dual-Arm Robot Manipulations". In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2341–2353. DOI: 10.1109/TRO.2021.3139838.

[17] Xiao Ma, David Hsu, and Wee Sun Lee. "Learning Latent Graph Dynamics for Visual Manipulation of Deformable Objects". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 8266–8273. DOI: 10.1109/ICRA46639.2022.9811597.

[18] Sotiris Makris, Emmanouil Kampourakis, and Dionisis Andronas. "On deformable object handling: Model-based motion planning for human-robot co-manipulation". In: *CIRP Annals* 71.1 (2022), pp. 29–32. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2022.04.048. URL: https://www.sciencedirect.com/science/article/pii/S0007850622000956.

[19] Changhao Wang Mingrui Yu Kangchen Lv. *A Coarse-to-Fine Framework for Dual-Arm Manipulation of Deformable Linear Objects with Whole-Body Obstacle Avoidance*. 2023. arXiv: 2209.11145 [cs.RO].

[20] Giorgio Nicola, Enrico Villagrossi, and Nicola Pedrocchi. "Human-robot co-manipulation of soft materials: enable a robot manual guidance using a depth map feedback". In: *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 2022, pp. 498–504. DOI: 10.1109/RO-MAN53752.2022.9900710.

[21] Gianluca Palli. "Model-based Manipulation of Deformable Linear Objects by Multivariate Dynamic Splines". In: *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*. Vol. 1. 2020, pp. 520–525. DOI: 10.1109/ICPS48405.2020.9274730.

[22] Agniva Sengupta et al. "Simultaneous Tracking and Elasticity Parameter Estimation of Deformable Objects". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 10038–10044. DOI: 10.1109/ICRA40945.2020.9196770.

[23] Te Tang, Changhao Wang, and Masayoshi Tomizuka. "A Framework for Manipulating Deformable Linear Objects by Coherent Point Drift". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3426–3433. DOI: 10.1109/LRA.2018.2852770.

[24] Yunxi Tang et al. "Learning-Based MPC With Safety Filter for Constrained Deformable Linear Object Manipulation". In: *IEEE Robotics and Automation Letters* 9.3 (2024), pp. 2877–2884. DOI: 10.1109/LRA.2024.3362643.

[25] Angel J. Valencia, Félix Nadon, and Pierre Payeur. "Toward Real-Time 3D Shape Tracking of Deformable Objects for Robotic Manipulation and Shape Control". In: *2019 IEEE SENSORS*. 2019, pp. 1–4. DOI: 10.1109/SENSORS43011.2019.8956623.

[26] Yuxuan Yang, Johannes A. Stork, and Todor Stoyanov. "Online Model Learning for Shape Control of Deformable Linear Objects". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 4056–4062. DOI: 10.1109/IROS47612.2022.9981080.

[27] Yuxuan Yang, Johannes A. Stork, and Todor Stoyanov. "Online Model Learning for Shape Control of Deformable Linear Objects". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 4056–4062. DOI: 10.1109/IROS47612.2022.9981080.

[28] Mingrui Yu, Hanzhong Zhong, and Xiang Li. *Shape Control of Deformable Linear Objects with Offline and Online Learning of Local Linear Deformation Models*. 2022. arXiv: 2109.11091 [cs.RO].

[29] Manuel Zürn et al. "Kinematic Trajectory Following Control For Constrained Deformable Linear Objects". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. 2021, pp. 1701–1707. DOI: 10.1109/CASE49439.2021.9551613.