Single Shot Learning of the Stiffness Distribution of a Soft Object

Lars Besselaar and Cosimo Della Santina

***Abstract*— With robotics rapidly expanding towards new user-oriented applications, such as agriculture, households, and classrooms, new tasks and new requirements arise. One of the ongoing and unsolved problems that comes with the unpredictable new environments robots find themselves in, is soft object manipulation. Without assuming the handled object to be rigid, its behavior under forces is no longer known: it could break, bend, plastically deform, or be crushed. In this paper, we propose a method for the identification of the deformation model of a soft object. We consider a bimanual robotic system manipulating a deformable object with the intention of controlling the object's shape. Since the object's deformation parameters are not known in advance, we propose a learning algorithm for their identification, and we specifically focus on learning the stiffness distribution along the object. By using a state space model based on modes of curvature, rather than a discrete Cartesian state space, we are able to learn the stiffness distribution of the deformable beam with only a single experiment. Then, we prove that the proposed method provides full control over the shape of the beam by performing a control task in simulation. Given the promising accuracy of the method, this work provides a solid foundation for future work in the direction of the fast identification of deformation parameters.**

I. Introduction

As the playing field of robotics expands from the controlled environment of manufacturing towards the unpredictable world of user-oriented applications, its need for adaptability and compliance increases. With the introduction of robotics in every-day household appliances [1], classrooms [2], and agriculture [3], the need for more intelligent robots increases. One of the relevant and ongoing problems that comes with the introduction of robotics into a wider span of environments is soft object manipulation. The applications of robotic manipulators that require dynamically manipulating deformable objects span from small, simple motions, such as as bending flower stems for cutting [4] or harvesting orchard crops [5], to large and challenging motions such as folding a table cloth [6] or spinning pizza dough [7].

The main challenge that arises from the manipulation of soft objects is the identification of the object's dynamics. If the complete dynamic behavior of a deformable object were to be known, a robotic manipulator would always be able to predict the object's deformation under application forces. This can already be done for, for instance, ideal theoretical beams; their dynamics are completely known, due to many simplifications and approximations. However, in real life, deformable objects are much more complex due to heterogeneity, nonlinearity, and geometric complexity. Rarely are deformable objects simple enough to be approximated by existing deformation models, and thus the need grows for more general approaches towards obtaining a soft object's dynamics.

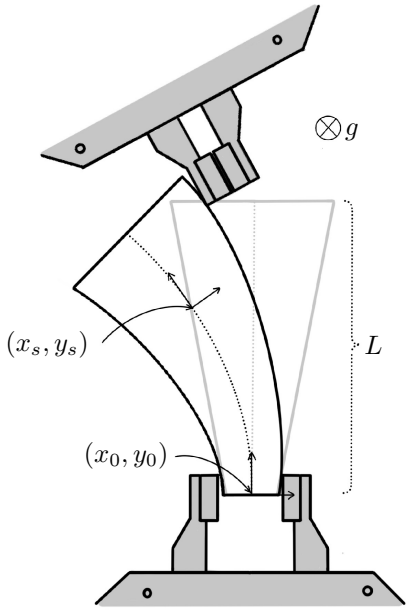


Fig. 1: Schematic overview of the bimanual robot and the soft object it manipulates. Here, (x_0, y_0) are the Cartesian coordinates of the beam at $s = 0$, (x_s, y_s) are the Cartesian coordinates at $s = 1$, g is the gravity and L is the length of the beam.

In the last few years, plenty of approaches have been developed with the intent of manipulating an unknown deformable object. Because there is no a priori information on the deformation behavior of the object at hand, two possibilities arise; firstly, one can learn to grasp and manipulate the object by trial and error, as done by Lin et al. [8]. However, in situations where the manipulated object is more delicate, one might not be able to afford such errors, due to the risk of breaking or plastically deforming the object. Therefore, secondly, methods have been developed for the estimation of the deformation of a deformable object. Using only kinematic information, this has been done by, for instance, Jordt et al. [9]. The authors use an RGB-D camera to observe the deformation of the object and continually update a 3D model to match the incoming visual data in order to obtain their deformation data. Alternatively, the physical and dynamical properties of the deformable object can be used to obtain certain deformation parameters. By assuming certain properties, such as considering beams as Euler-Bernoulli beams or Kirchoff rods, methods have been developed [10] in which the Young's modulus and Poisson's ratio can be

obtained using *analysis by synthesis*. Additionally, work has been done to combine deformable object manipulation with neural networks. In these approaches, the manipulator's end-effector movement is mapped to the soft object's deformation [11], [12], eliminating the need for explicit modeling at the cost of a need for large amounts of data.

Parallel to the field of deformable object modeling, the field of soft object control has been developing many methods that are suitable for transfer between the fields. Many researchers have implemented neural networks in the control loops of their soft robots [13], [14], [15], in order to successfully and accurately actuate their soft robots. Alternatively, adaptive control loops have been used for soft robots that operate under some uncertainty; pertaining to, for instance, the variable stiffness of the robot arm [16], a changing environment [17], or unknown robot parameters [18].

One of the problems that remains in the state of the art is the application of the obtained deformation model to the manipulation and control of the object. While the mentioned works have provided important and valuable improvements in their respective fields, most works only focus on the deformation model itself, without showing how their model can be applied to a manipulation task. The works that do explicitly focus on manipulation tasks use a learning strategy, such as adaptive control or neural networks, thus removing the intuitiveness that comes with having an analytical model. Other works that focus on actual control are the works from the field of soft robotics; however, in this field, the parameters of the robot are often known in advance, contrary to the unknown soft object of interest.

In this paper, we combine the advancements made in soft robotics with the need for deformable object models that are easily applicable for control. In previous work, Della Santina [19] and Rus [20] developed a curvature based deformation model, intended for the control of their soft robot. This model has great potential to be adopted for the control of an unknown soft object that resembles a slender beam. However, we do not know any of the deformation parameters that are required to complete the deformation model; especially the stiffness distribution along the slender beam. Therefore, we propose a learning strategy to obtain the stiffness distribution along the length of the beam. Once the stiffness function has been obtained and the deformation model is fully defined, we apply the model in a simulated environment to show that we now have full control over the deformation of the slender beam.

II. The Soft Object Model

We consider a planar slender beam of length L and thickness D. A schematic figure of the beam can be found in Figure 1. Between the base of the beam and the tip of the beam, we introduce a normalized coordinate $s \in [0, 1]$, with $s = 0$ at the fixed base of the beam, and $s = 1$ at the tip of the beam. The beam is considered to be inextensible, so the length of the beam remains L as it is being deformed. Additionally, we assume a constant density ρ along the beam, but a non-constant stiffness distribution $k(s)$.

A. Kinematics

If we were to assume that the beam would have a constant curvature along its length, we could fully specify the shape of the beam using a single value for the beam's curvature. This concept can be expanded by extending the curvature function from constant to affine; now, two values dictate the shape of the beam: one constant term, and one curvature term that describes a linear relationship between s and the curvature. Mathematically, this would look like;

$$q(s,t) = \theta_0 + \theta_1 s, \tag{1}$$

where $q(s,t)$ is the curvature function, and θ_0 and θ_1 are the constant and linear curvature coefficients, respectively. Continuing this trend, we can fully define any beam by means of an infinite summation of monomials;

$$q(s,t) = \sum_{i=0}^{\infty} \theta_i(t) s^i. \tag{2}$$

Using this relationship between the normalized coordinate s and the curvature function $q(s,t)$, we can reduce the size of the state space by using a finite amount of monomials rather than an infinite sum. This way we can approximate the pose of the rod using n curvature modes, rather than some spacial discretization in the Cartesian state space. Additionally, we can easily convert a modal state space to the local orientation and subsequently the Cartesian coordinates of a point along the rod by first integrating the curvature function $q(s,t)$ to obtain the angle function;

$$\alpha(v,t) = \int_0^v q(s,t)\mathrm{d}s = \sum_{i=0}^{\infty} \theta_i(t) \frac{v^{i+1}}{i+1}, \tag{3}$$

and then integrating the sine and cosine of the angle function to obtain the y and x coordinates, respectively;

$$\frac{x(v,t)}{L} = \int_0^v \cos(\alpha(s,t))\mathrm{d}s, \quad \frac{y(v,t)}{L} = \int_0^v \sin(\alpha(s,t))\mathrm{d}s. \tag{4}$$

B. Dynamics

The dynamics of the slender rod expressed in the curvature modes are derived in the same manner as [21]. Thus, for the sake of brevity, the derivations of the inertia matrix, the Coriolis and centrifugal terms, the input field, and the damping forces will be omitted in this paper. The elastic field of the beam, however, is used in the learning strategy that will be proposed in the next sections. For this reason, its derivation will be shown explicitly.

The elastic field is obtained by defining the total energy stored in the rod, and then taking the partial derivative of the total energy with respect to each mode. In the case of a rod with a variable stiffness function $k(s)$ along the length of the rod, the total energy can be expressed as;

$$U_E(t) = \int_0^1 \frac{k(s)}{2} q^2(s,t)\mathrm{d}s. \tag{5}$$

Then, by evaluating the partial derivative with respect to θ_i, we arrive at;

$$\begin{aligned} G_{E,j} &= \int_0^1 k(s) \frac{\partial}{\partial \theta_k} \frac{1}{2} q^2(s,t) \mathrm{d}s \\ &= \int_0^1 k(s) q(s,t) \frac{\partial}{\partial \theta_k} q(s,t) \mathrm{d}s \\ &= \int_0^1 k(s) \left(\sum_{i=0}^{\infty} \theta_i(t) s^i \right) s^j \mathrm{d}s \\ &= \sum_{i=0}^{\infty} \theta_i(t) \int_0^1 k(s) s^{i+j} \mathrm{d}s. \end{aligned} \tag{6}$$

Having arrived at an expression for the elastic field in terms of the modes of curvature, we can see Equation (6) as a matrix multiplication $K\Theta$, with K being the square stiffness matrix, equal to the integral of $k(s)s^{i+j}$, and Θ equaling a vertical array of modal states.

Combining the findings of the elastic field with the derivations of the other components of the dynamical system as in [21], we find the following complete dynamical system:

$$B(\Theta)\ddot{\Theta} + C(\Theta, \dot{\Theta}) + G_G(\Theta, \phi) + K\Theta + D\dot{\Theta} = A\tau, \tag{7}$$

where B is the inertia of the rod, C is the Coriolis and centrifugal terms, G_G is the gravitational field, K is the previously derived stiffness matrix, D is the damping matrix, and A is the input field that transforms applied forces and torques to the force and torque acting on each mode.

C. Assumptions and simplifications

In order to isolate the effects of the stiffness profile of the deformable beam from the dynamical system (7), we simplify the motion of the beam under perturbation. Firstly, we assume quasi-static motion, which allows us to assume the first and second derivative of the modal state to be zero. Secondly, the beam will only be perturbed in the plane orthogonal to the gravitational field, causing gravity's effect on the motion of the perturbed beam to be negligible. Following these two simplifications, we can eliminate the inertial, centrifugal, gravitational and damping terms from the dynamical system. The dynamics are now;

$$K\Theta = A\tau. \tag{8}$$

We now rewrite the left side of Equation (8) to obtain an expression that explicitly contains the stiffness function. To do so, we acknowledge that $K\Theta$ is an alternative expression for the elastic force field G_E. Thus,

$$\sum_{i=0}^{\infty} \theta_i \int_0^1 k(s)\, s^{i+j} \mathrm{d}s = A\tau. \tag{9}$$

Lastly, the applied torque on the right hand side of Equations (7), (8) and (9) will be replaced with an applied force, since an applied force is much easier to apply and measure. To perform this replacement, the following relationship is used;

$$\tau = J(s_a)^\top F_a, \tag{10}$$

where $J(s_a)^\top$ is the transpose of the Jacobian, evaluated at the point of application in terms of s. Since the Jacobian describes the relationship between the rate of change of Cartesian states and modal states, we do not need the transformation matrix A that was previously used to obtain the torque expressed in modal coordinates. The final dynamic relationship to be used for the estimation of the stiffness function is thus;

$$\sum_{i=0}^{\infty} \theta_i \int_0^1 k(s)\, s^{i+j} \mathrm{d}s = J(s_a)^\top F_a. \tag{11}$$

III. Learning Algorithm

A. Approximation Functions

We've now arrived at a notation that describes a relationship between applied forces, observed beam states, and the stiffness function. Since the goal of the learning algorithm is to regress a continuous stiffness distribution, we need to introduce a family of approximation functions to parametrize $k(s)$. Once parametrized, we solve for the parameters of the approximation function to obtain the sought stiffness function.

The simplest family of approximation functions is the polynomial family, whose coefficients serve as the parameters to be learned. A second, more complex option is to use a sum of m amount of Gaussians, placed at equidistant fixed locations between $s = 0$ and $s = 1$. In this context, the heights of the Gaussians are the parameters to be learned. Each family has an advantage over the other: polynomials have the advantage of being computationally inexpensive due to the trivial integral that arises with the substitution of $\sum_{w=0}^{m} \alpha_w s^w$ into Equation (9), and sums of Gaussians have the advantage of being less restrictive in shape, allowing for a more nuanced approximation.

However, when both of the mentioned approximation functions are given the same amount of free parameters (for example, a second order polynomial and a sum of three Gaussians) the Gaussians will approximate the same function shape as the polynomial. Likewise, the sum of two Gaussians will approximate an affine function, and the sum of four Gaussians will approximate a cubic function. However, the polynomial approximation function has the disadvantage of growing towards $\pm\infty$ beyond $s \in [0, 1]$, while Gaussians can take constant values near $s = 0$ and $s = 1$. For this reason, only Gaussian approximation functions will be considered from this point onwards.

B. Linear Regression

First, we evaluate the resulting expression when substituting a general polynomial into Equation (9):

$$\sum_{i=0}^{\infty} \theta_i \int_0^1 \sum_{w=0}^{m} \alpha_w s^w\, s^{i+j} \mathrm{d}s = J(s_a)_j^\top F_a. \tag{12}$$

Since the polynomial coefficients are not functions of s, we can separate these terms:

$$\sum_{w=0}^{m} \alpha_w \sum_{i=0}^{\infty} \theta_i \int_0^1 s^{i+j+w} \mathrm{d}s = J(s_a)_j^\top F_a. \tag{13}$$

And, when solving the definite integral, it becomes apparent we have created a linear relationship between the states θ, the stiffness function parameters α, and the applied torque τ.

$$\sum_{w=0}^{m} \alpha_w \sum_{i=0}^{\infty} \frac{\theta_i}{i+j+w+1} = J(s_a)_j^\top F_a. \tag{14}$$

Similarly, we can substitute the function template for a Gaussian distribution into Equation (9):

$$\begin{gathered} \sum_{i=0}^{\infty} \theta_i \int_0^1 \sum_{w=0}^{m} \beta_w \Pi_{G,w}(s)\, s^{i+j} \mathrm{d}s = J(s_a)_j^\top F_a, \\ \text{where} \quad \Pi_{G,w} = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{s-\mu}{\sigma}\right)^2} \end{gathered} \tag{15}$$

where each Gaussian $F_{G,w}$ is centered at a different s-coordinate, and β_w denotes the weight of each Gaussian. We can now, again, separate the weights from the integral;

$$\sum_{w=0}^{m} \beta_w \sum_{i=0}^{\infty} \theta_i \int_0^1 \Pi_{G,w}(s)\, s^{i+j} \mathrm{d}s = J(s_a)_j^\top F_a, \tag{16}$$

but, in the case of the Gaussian approximation function, the integral cannot be as compactly evaluated as in the case of the polynomial approximation function. The linear structure of the regression, however, will be exactly the same. In matrix form, the general problem looks as follows;

$$\sum_{i=0}^{n} \theta_i \begin{bmatrix} \int_0^1 \Pi_a(s)\, s^{i+j} \mathrm{d}s \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \vdots \\ \gamma_m \end{bmatrix} = \begin{bmatrix} J(s_a)_j^\top F_a \end{bmatrix}, \tag{17}$$

where γ is a generic function parameter and Π_a is a generic approximation function. In Equation (17), the w-index increases with each column, the j-index increases with rows, and the i index serves as a third dimension over which we sum each matrix after multiplying with θ_i. We can now obtain the approximation function parameters γ by inverting Equation (17). For brevity, we write Equation (17) as $X\gamma = Y$, where $X = \sum_{i=0}^{n} \theta_i \int_0^1 \Pi_a(s)\, s^{i+j} \mathrm{d}s$ and $Y = J(s_a)_j^\top F_a$. Since X and Y have as many columns as observations, and can therefore be assumed to not be square, we need to invert (17) using a pseudo-inverse. Thus, we find

$$\gamma = (X^\top X)^{-1} X^\top Y, \tag{18}$$

which results in a least-squares solution for the function parameters γ of the stiffness function.

C. Hyperparameter Selection

When considering Gaussian approximation functions, the two contributing factors to the accuracy of the stiffness function estimation are the number of modes n used to approximate the beam shape, and the number of equidistant Gaussians placed along the length of the rod m. Since the optimal values for these two parameters is not immediately obvious, a hyperparameter optimization is performed, based on the Bayesian Information Criterion (BIC):

$$BIC = k\ln(N) - A\ln(L). \tag{19}$$

The BIC acts as a measurement of model quality, and considers the accuracy of the model while penalizing the complexity of said model. In (19), k is the number of parameters of the model, N is the number of observations, A is a weighing constant and L is, in this application, the error of the model.

In the context of this research, k is the sum of n and m; N is the amount of data points collected during an experiment; and L is the error of the model, which will be defined in Section IV.

Firstly, we define a maximum amount of modes z that can be evaluated. Given that we observe the shape of the beam through positional tags, as will be discussed in Section IV-A, we can only evaluate as many modes as there are observations due to the inverse kinematics. Since each tag gives information on both the x-coordinate and the y-coordinate of a specific point, the upper limit on z is two times the amount of tags.

Due to the nature of the linear regression, the problem becomes underdefined when $m > n$. Therefore, we iterate over $m \in [2, z]$ for each value of n. We exclude $m = 1$ because a sum of a single Gaussian would just be another Gaussian, which is too limiting as an approximation function. Consequently, we exclude $n = 1$, since the only valid value for m in this configuration would be $m = 1$.

Finally, we obtain the optimal values of n and m by analyzing the BIC of every model and choosing the model with the lowest BIC:

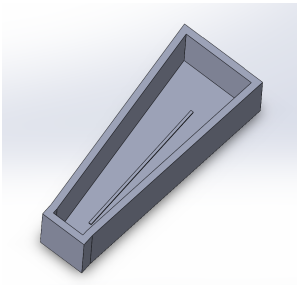
$$\begin{aligned} (\hat{n}, \hat{m}) = &\underset{n,m}{\operatorname{argmin}}\, k\ln(N) - A\ln(L) \\ &n \in [2, z] \\ &m \in [2, n]. \end{aligned} \tag{20}$$

This optimization will be performed on the training data, after which the obtained stiffness function will be used for the validation data to verify the results.

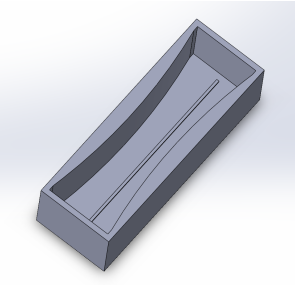
IV. Experiments

First, the experiment setup will be shown, and the reasoning behind certain design choices will be discussed. Then, since the process required to transform real-life experiment data to modal states and forces is not trivial, we will elaborate on the designed data pipeline. Finally, the perturbation strategies will be explained.

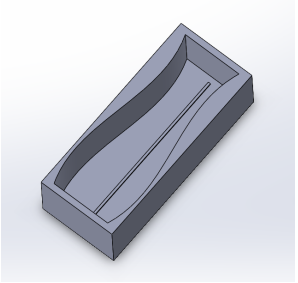
Fig. 2: Experiment setup. Visible are the Franka Emika Panda robot, the silicone beam, and the recording device attached to a tripod.



(a) Beam design with a linear width function.



(b) Beam design with a quadratic width function.



(c) Beam design with a cubic width function.

Fig. 3: 3D-models for the molds of the beam designs.

A. Experiment Setup

Since the relationship we last arrived at, shown in Equation (8), relates the modal states and the stiffness function to the applied forces, we will need an appropriately large set of data containing both positional and force data. To obtain this dataset, an experiment setup has been designed in which the Franka Emika Panda robot acts as the manipulator. In the experiment, a silicone beam will be secured on one end, while the robot applies a force on the other end. While this force is being exerted, a camera setup records the motion of the perturbed beam using AprilTags [22]. AprilTags are small two-dimensional barcodes for which a robust library of detection software is built, which allows for spacial tracking of the tags using any ordinary recording device. By recording the forces that are being exerted onto the beam, as well as the deformation of the beam as an effect of the applied forces, we can build the dataset we need for the regression of the stiffness function.

B. Beam Design

The design of the beam is an essential factor for the successful execution of experiments. Firstly, the choice of material needs to be correct; if the material is too stiff, the manipulating robot will not be able to exert the required forces to deform it, and if the material is not stiff enough, the forces required for deformation will be too small to be accurately read. Additionally, if the material is too hard, the beam might break before it shows any significant deformation at all. After some trial-and-error, choosing a pourable silicon with a hardness of 30 on the Shore hardness scale proved to be the optimal choice.

Secondly, the geometric design of the beam is of importance. While the ideal way of creating a beam with variable stiffness would be to use silicones of varying stiffness along the length of the beam, this is highly impractical. A more practical alternative is to vary the width of the beam along its length, since the bending stiffness of a beam is dependent on its moment of inertia, thus its cross-sectional area, according to the Euler-Bernoulli equation;

$$M = EI\frac{\mathrm{d}^2 w}{\mathrm{d}w^2}. \tag{21}$$

Therefore, three different beam designs have been created, each with a unique stiffness function along its length. In Figure 3, the molds for each of the three beam shapes can be seen. The intention behind the designs of the beams is two-fold: firstly, each beam has a width function modeled after a unique order of polynomial. This is done so that the linear, quadratic and cubic beam should each yield a stiffness function of increasing polynomial complexity. Secondly, the beams have been designed so that the point along the rod where the stiffness function is lowest is different for each beam. The linear beam, seen in Figure 3a, has its structurally weakest point at its base, the quadratic beam as seen in Figure 3b has its weakest point halfway along the length of the beam, and the cubic beam as seen in Figure 3c has its weakest point near the tip of the beam. This design choice has been made to ensure that the motion of the beams under deformation will differ from one another as much as possible, providing a good foundation for the validation of the method.

C. Data Pipeline

Once the experiments have been performed, a significant amount of processing is required to prepare the gathered data for the linear regression. The data pipeline can be divided into three subsections: obtaining positional data through tag detection, timestamp matching, and inverse kinematics. Positional data is obtained through the use of four AprilTags:

one mounted at the base, acting as the origin, and three along the length of the beam, roughly equally spaced at $s = \frac{1}{3}$, $s = \frac{2}{3}$, and $s = 1$. By analyzing each frame of the recorded video, we can determine the relative positioning of the three tags with respect to the origin tag in terms of pixels. Then, by comparing the real-life size of a tag with the pixel-size of a tag, we can establish a millimeters-per-pixel ratio that we can use to transform the pixel distances to Cartesian distances.

Additionally, since we need pairs of states and forces at each timestep, a timestamp needs to be assigned to each measurement of Cartesian distances. By storing the exact starting point of the video, given that the frame rate of the video is known and constant, we can obtain the timestamp of each frame using $t = t_{\text{start}} + nf$, where t_{start} is the stored starting time, n is the number of frames that have passed since the first, and f is the frame rate of the video. Similarly, the force exerted by the robot is also stored along with timestamps. However, because the frequencies at which the robot stores data is not equal to the frame rate at which the video records, and the starting times are not equal due to both recording media being started manually, the data needs to be aligned after the experiment. This is done by looping over both the force data and the positional data simultaneously, and storing a data pair when the difference between the timestamps of two measurements is smaller than some margin of error ϵ.

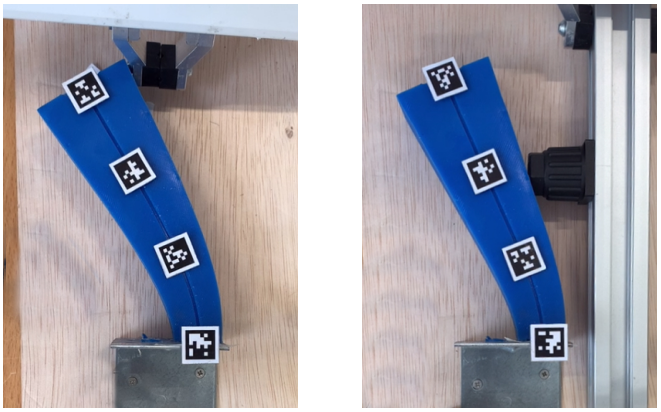
Lastly, since all dynamics are expressed in modal states θ, the Cartesian positional data needs to be converted to modal states. This is done using the Newton-Raphson method;

$$\theta_{k+1} = \theta_k + J^{+}(y_m - y_g)_k, \tag{22}$$

where θ is the set of modal states, k is the iteration step, J is the Jacobian, y_m are the measured marker positions, and y_g is the transformation of the modal states to Cartesian coordinates at $s \in [\frac{1}{3}, \frac{2}{3}, 1]$ using Equation (4). This algorithm allows us to use the positional data of the three recorded markers simultaneously to find the set of modal states that minimizes the error between the virtual beam and the experimental markers. As a result, we now have a timestamp-matched set of force-state pairs, which can be used for the linear regression problem that was created in Equation (17).

D. Perturbation Strategies

In order to obtain a reliable insight into the effectiveness of the method, both a set of training data as well as a set of validation data will be gathered. The training data will be a single experiment in which the beams are perturbed at their tips in a slow and controlled fashion. An example of a beam being perturbed at its tip can be found in Figure 4a. For the validation data, three different experiments will be executed: firstly, one in which the perturbation trajectory is executed 2 to 3 times as fast, which means that fewer data points will be collected in the timespan of the experiment. Secondly and thirdly, experiments will be executed where the robot applies a force at $s = 0.8$ and $s = 0.6$. An example of a beam being perturbed at a lower value of s can be found in Figure 4b. A consequence of applying the force away from the tip is that information is lost on the deformation behavior near the tip of the beam. The limitations implemented in the validation experiments are designed with the intent of testing whether the stiffness function obtained with the training data is accurate enough to support the manipulation of the beams in other ways than the one used in the training experiments.



(a) Linear beam being perturbed at $s = 1$.

(b) Linear beam being perturbed at $s = 0.6$.

Fig. 4: Frames of the captured video data.

E. Error definition

In order to be able to compare and validate results, we need to define a measure of quality, or, more specifically, a measure of error. We can do so by comparing the motion of the experiment beam with the motion of a simulated beam that is assigned the estimated stiffness function. The quality of the estimated stiffness function can then be defined by the error between the motion of the experiment beam and the motion of the simulated beam. In Figure 5, the order of operations can be seen for the the assessment of the quality of the stiffness function. Although the most straightforward comparison seems to be between the modal states of the experiment beam and the simulated beam, this is not the most accurate comparison. This is because the modal states of the experiment beam are an approximation of true modal states of the experiment beam, derived from the observed tags. Therefore, the modal states might be more accurate for $n = 5$ than for $n = 2$, which in turn means that for low values of n, the modal states of the simulated beam might be "equally bad" as the modal states of the experiment beam, resulting in a deceivingly high accuracy. It is therefore more reliable to compare the true observations, the observed tags, with specific points along the simulated beam. We will therefore define the accuracy of the hyperparameter values as the RMS of the difference between the observed tags and the Cartesian coordinates over time, obtained at the same s-coordinates along the simulated beam.

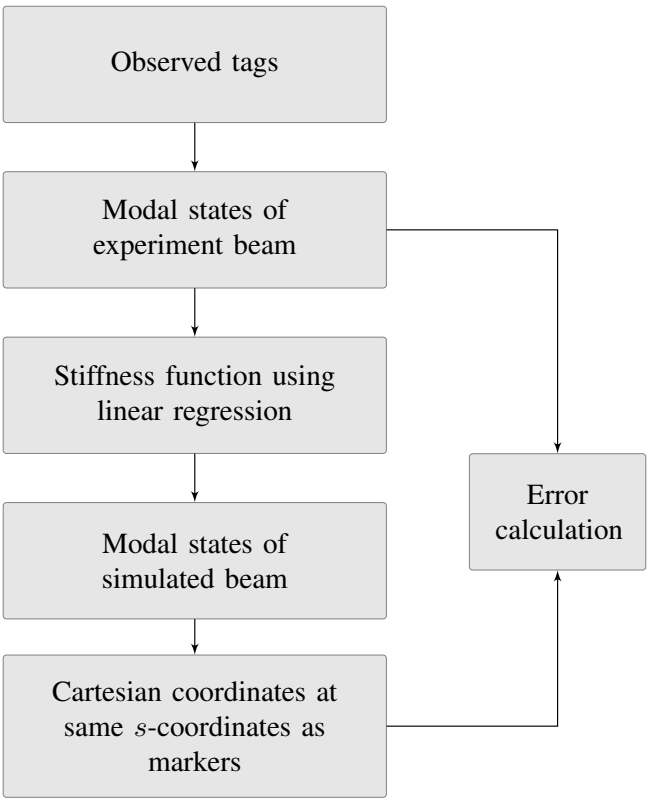


Fig. 5: Order of operations for the assessment of the quality of the stiffness function.

V. Results

To show the effectiveness of the method, the performance of the algorithm on the training data will first be shown. Then, the validation sets are used to analyze whether the stiffness function estimation was successful.

A. Training Data

Performing the hyperparameter optimization on the training data yielded the error data as can be seen in Tables I, II an III. For brevity, the beam with a linear width function will be referred to as the linear beam, and similarly for the quadratic and the cubic beams. During the hyperparameter optimization, we iterate through $n \in [2, z]$, where we limit z at 5 due to computational limitations at higher amounts of modes.

The results for the linear beam, as seen in Table I, vary the most in quality. The optimal combination of hyperparameter is therefore very clearly $n = 4, m = 4$. Using this set of hyperparameters to estimate the stiffness function yields the stiffness function as shown in Figure 6a. Here, a clear section of low stiffness can be seen between $s = 0$ and $s = 0.3$,

TABLE I: RMS of errors of the linear beam.

Params	M_1 [m]	M_2 [m]	M_3 [m]	BIC
$n = 2, m = 2$	0.0026	0.0026	0.0204	6.13
$n = 3, m = 2$	0.0040	0.0071	0.0070	5.47
$n = 3, m = 3$	0.0035	0.0087	0.0367	6.73
$n = 4, m = 2$	0.0051	0.0044	0.0155	5.92
$n = 4, m = 3$	0.0014	0.0050	0.0067	5.23
$\mathbf{n = 4, m = 4}$	**0.0025**	**0.0029**	**0.0034**	**4.74**
$n = 5, m = 2$	0.0047	0.0082	0.0059	5.51
$n = 5, m = 3$	0.0024	0.0068	0.0096	5.59
$n = 5, m = 4$	0.0026	0.0063	0.0045	5.20
$n = 5, m = 5$	0.0029	0.0070	0.0039	5.29

TABLE II: RMS of errors of the quadratic beam.

Params	M1 [m]	M2 [m]	M3 [m]	BIC
$n = 2, m = 2$	0.0012	0.0010	0.0022	4.10
$n = 3, m = 2$	0.0013	0.0011	0.0021	4.10
$n = 3, m = 3$	0.0012	0.0011	0.0021	4.09
$n = 4, m = 2$	0.0016	0.0014	0.0020	4.16
$n = 4, m = 3$	0.0013	0.0013	0.0021	4.13
$\mathbf{n = 4, m = 4}$	**0.0010**	**0.0014**	**0.0021**	**4.08**
$n = 5, m = 2$	0.0010	0.0012	0.0023	4.12
$n = 5, m = 3$	0.0025	0.0023	0.0036	4.69
$n = 5, m = 4$	0.0011	0.0023	0.0030	4.45
$n = 5, m = 5$	0.0059	0.0129	0.0228	6.39

TABLE III: RMS of errors of the cubic beam.

Params	M1 [m]	M2 [m]	M3 [m]	BIC
$n = 2, m = 2$	0.0010	0.0035	0.0066	5.11
$\mathbf{n = 3, m = 2}$	**0.0013**	**0.0030**	**0.0066**	**5.09**
$n = 3, m = 3$	0.0203	0.0450	0.0682	7.53
$n = 4, m = 2$	0.0011	0.0040	0.0065	5.14
$n = 4, m = 3$	0.0014	0.0057	0.0078	5.38
$n = 4, m = 4$	0.0195	0.0390	0.0695	7.50
$n = 5, m = 2$	0.0010	0.0035	0.0065	5.10
$n = 5, m = 3$	0.0010	0.0036	0.0065	5.12
$n = 5, m = 4$	0.0126	0.0248	0.0369	6.93
$n = 5, m = 5$	0.0095	0.0205	0.0348	6.82

after which the stiffness increases. Near the end of the beam, it can be seen that the stiffness of the beam decreases again, which contradicts with the original hypothesis of correlation between beam width and stiffness. This counter-intuitive phenomenon can be explained by the high sensitivity for inaccuracies the algorithm has near the tip of the beam.

The estimation of the stiffness function for the quadratic beam has a very high quality for almost every combination of hyperparameters, as can be seen in Table II. From the calculation of Q, it can be seen that the hyperparameters $n = 4, m = 4$ have a marginally better BIC than the rest of the hyperparameter combinations. Additionally, the stiffness function that is produced using these hyperparameters follows the width function along the length of the beam well: the stiffness is high at the base and the tip, and low in the middle of the beam.

Lastly, the quality index for the cubic beam is found at

$n = 3, m = 2$, which is a combination of hyperparameters of much lower complexity than the quadratic and linear beam. The estimated stiffness function using these hyperparameters can be found in Figure 3c, where it can be seen that the stiffness is high around $s = 0.25$, and decreases afterwards. This function follows the shape of the cubic beam well.

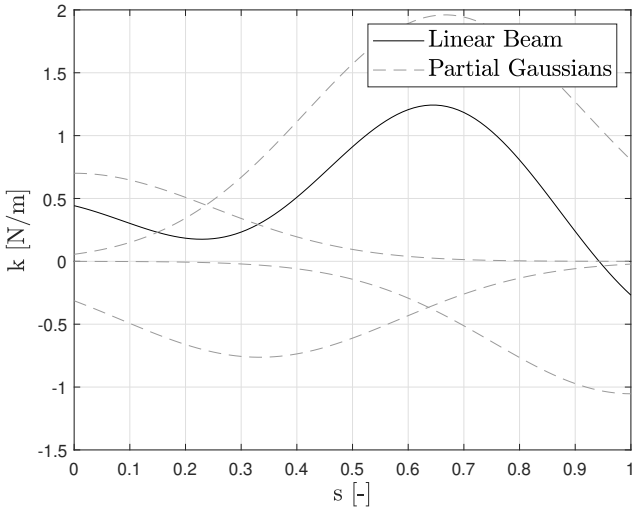
From Tables I, II, III, it can be seen that the magnitude of the error is in the order of millimeters. Generally, the error is largest at the third marker, which is placed at the tip of each beam. This can be explained by the fact that the tip of the beam generally experiences the largest deflection when a force is applied to the tip. However, although the low errors for the training data are a positive results, no conclusions on the accuracy of the found stiffness distributions can be drawn before some proper validation tests.

B. Validation

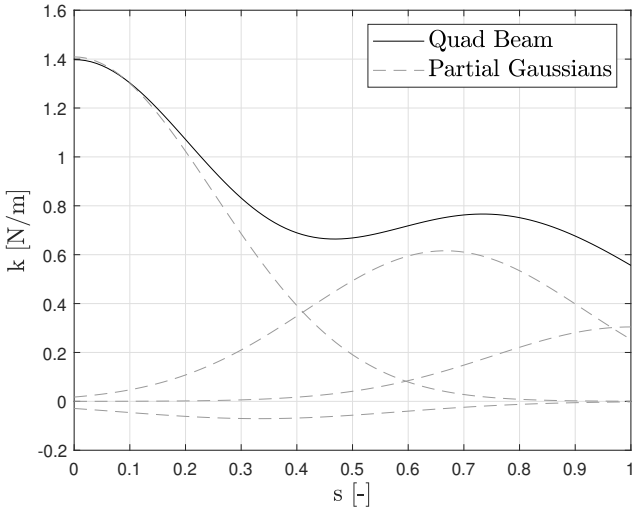
As explained in Section IV-D, we aim to validate the results that were found using the training data by using experiment data collected with different perturbation strategies. For these validation sets, we use the stiffness distributions found using the training set, assign these stiffness distributions to a simulated beam, and apply the same forces as measured from the validation experiments to analyze the error between the simulated and experiment beams once again. We do so to verify that the found stiffness distributions are not overfitted to the training data, but are indeed accurate estimations of the stiffness distributions of the true beams. In Tables IV, V, and VI, the results of these comparisons can be found.

It can be seen that for all experiments, the error measurements for each marker remained in the order of millimeters, confirming that the stiffness distributions found using the training data are generally valid and not overfitted. Some bias was introduced, however, in the measurements for the experiments using lower points of application for the force in case of the quadratic and the cubic beam. Because these beams are stiff near the base, a disproportionately large force was needed in order to produce enough motion, which the robotic manipulator was not able to provide. Therefore, the motion of the beam during these experiments was very shallow, whereas most of the error occurs at large deflections. As a consequence, the error is inherently lower compared to an experiment in which a large deflection was achieved; however, since the errors are low for all validation experiments, we can assume that the estimated stiffness distributions are accurate.

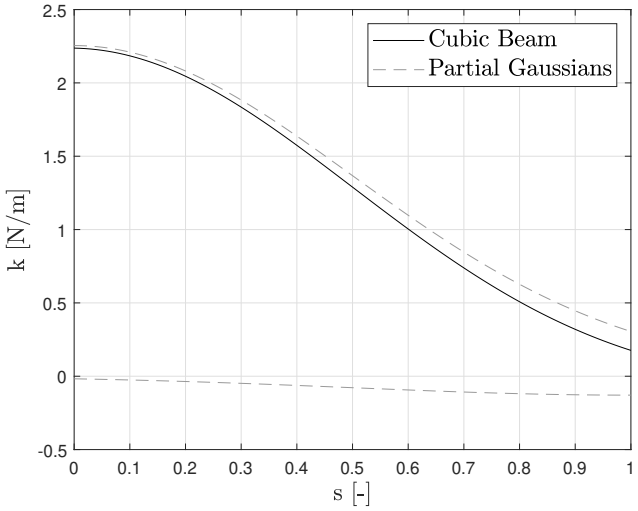
To provide a more intuitive insight into the accuracy of the method, the final positions of the experiment motions have been plotted for each beam, along with the position of the simulation beam that has been assigned the learned stiffness function. As can be seen in Figure 7, the shapes of the observed beams and the simulated beams match are very similar, implying that the stiffness function is accurate enough to obtain robust control over the deformation of the beam. To give an indication of the range of accuracies found in the validation experiments, the worst performance, best



(a) Estimated stiffness function of the linear beam.



(b) Estimated stiffness function of the quadratic beam.



(c) Estimated stiffness function of the cubic beam.

Fig. 6: Estimated stiffness function of each of the beam designs.

Algorithm 1: Iterative algorithm for determining the force required to move a specific point along the beam to a desired location.

while $\text{norm}(\Delta) > \epsilon$ **do**
  $\Theta_{k+1} = K^{-1} J_\Theta(s_a) F_{a,k}$
  $x_{\text{sim}} = h(\Theta_{k+1}, s_a)$
  $\Delta = x_{\text{goal}} - x_{\text{sim}}$
  $F_{a,k+1} = F_{a,k} + \gamma J_F^+(s_a)\Delta$
  $k = k + 1$
end

performance, and an example of an average performance have been plotted.

TABLE IV: RMS of errors: validation set of linear beam.

Strategy	M1 [m]	M2 [m]	M3 [m]
Fast	0.0008	0.0016	0.0038
$s_{app} = 0.8$	0.0002	0.0015	0.0035
$s_{app} = 0.6$	0.0014	0.0040	0.0069

TABLE V: RMS of errors: validation set of quadratic beam.

Strategy	M1 [m]	M2 [m]	M3 [m]
Fast	0.0005	0.0017	0.0039
$s_{app} = 0.8$	0.0005	0.0021	0.0049
$s_{app} = 0.6$	0.0008	0.0022	0.0033

TABLE VI: RMS of errors: validation set of cubic beam.

Strategy	M1 [m]	M2 [m]	M3 [m]
Fast	0.0012	0.0023	0.0039
$s_{app} = 0.8$	0.0006	0.0015	0.0032
$s_{app} = 0.6$	0.0005	0.0016	0.0032

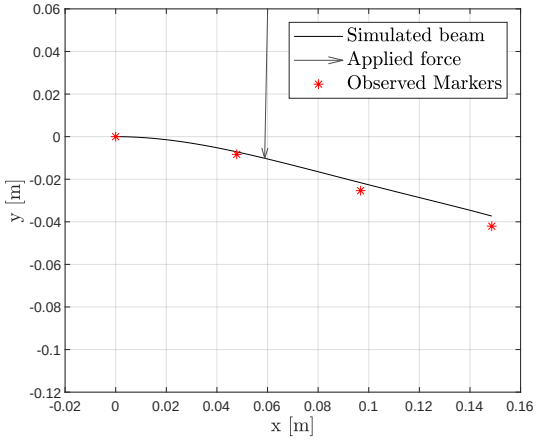
C. Manipulation task

To support the claim of being able to control the deformable beam using the obtained stiffness distribution, a set of manipulation tasks is performed in simulation. For each beam, we will show that, given a desired location for a specific point along the beam, we can calculate the force required for that location to reach its desired location.

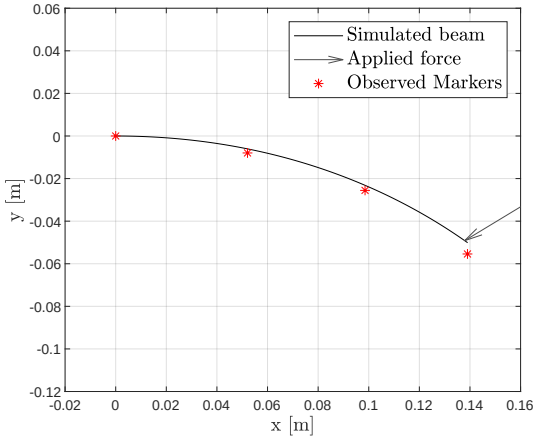
Assuming that the beam needs to be in equilibrium upon reaching its desired location, we can eliminate all derivatives from Equation (7) and arrive at

$$K\Theta = J_\Theta(s_a)^\top F_a, \tag{23}$$

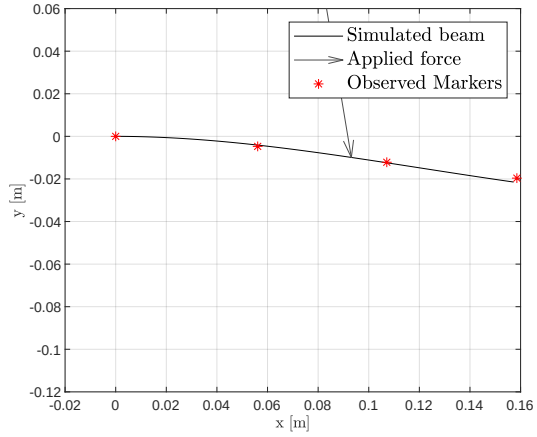
Where J_Θ is the Jacobian between the Cartesian and modal state descriptions, K is the stiffness matrix, s_a is a normalized location along the beam, and F_a is the applied force. However, we cannot directly solve for the applied force F_a, since the desired modal states are unknown. Although a single desired location is known, many combinations of modal states could place the point of interest at its desired location, and only one of those combinations can be reached



(a) The $s = 0.6$ strategy applied to the linear beam yielded the least accurate results.



(b) The fast strategy applied to the cubic beam yielded averagely accurate results.



(c) The $s = 0.6$ strategy applied to the cubic beam yielded the best results.

Fig. 7: Three results of the validation experiments are shown, where the measured markers are compared to a simulated beam that is assigned the learned stiffness distribution. To give an indication of the range of the quality of the results, the worst, best, and average results are shown.

by the beam due to its unique stiffness distribution. Therefore, an iterative approach needs to be implemented to find the applied force F_a that places the point of interest at its desired location. During the iterative approach, we aim to incrementally adjust the applied force in the direction that decreases the error between the point of interest and its desired location. Therefore, we set up a gradient descent method, for which we need the gradient from the applied force F_a and x. We can find this gradient by first creating an explicit relationship between Θ and F_a using (23):

$$\Theta = K^{-1} J_\Theta(s_a)^\top F_a. \tag{24}$$

Then, by using the forward kinematics as shown in Section II-A, we can define the point of interest x as $x = h(\Theta)$, with $h(\cdot)$ being the forward kinematics. The Jacobian of x with respect to F_a is then;

$$J_F = \frac{\partial h(\Theta)}{\partial F_a} = \frac{\partial h(K^{-1} J_\Theta(s_a)^\top F_a)}{\partial F_a}. \tag{25}$$

By implementing the found Jacobian J_F into the gradient descent method, we arrive at Alg. 1.

In total, 6 different manipulation tasks have been performed; two for each beam, where the first task revolves around the tip of the beam, and the second revolves around a point in the middle of the beam. The objective of both tasks is to move the point of interest to a desired location. In Figure 8, the final positions of each beam, along with periodic snapshots of the intermediate beam positions, can be seen. Although all manipulation tasks were performed with an error of 1 millimeter or less, these tasks are more so a prove of concept than to be used as a measure of accuracy. Using these results, the strategy for obtaining the required applied force can be transferred to a real life setting, where a measure of accuracy would be more valuable. However, due to restrictions in the capabilities of the robotic manipulator at hand, this could not be done during this research.

VI. Discussion

A. Performance

The results that were shown in Section V showed that the RMS of the errors ranged between 2 mm and 3 mm for the training data. Given that the force measurements were not as accurate as they could have been, it can be said that a portion of that error is due to a difference in force applied to the experiment beam and the simulated beam. When constantly applying the exact same force to both the experimental and the simulated beam, the errors between them ranged between 0.1 mm and 0.5 mm, while the length of the beams is 150 mm. The small size of these errors shows the accuracy of the estimated stiffness function. Additionally, the errors were comparably small for the set of validation experiments, indicating that the obtained stiffness functions in the training data were not overfitted, and are a good estimate of the true stiffness functions of the beams. For most manipulation tasks, performing within the shown error range is more than accurate enough, which allows us to conclude that the performance of the method is satisfactory.

B. Shortcomings

The main shortcoming in this method is the inaccuracies in the force measurement of the Franka Emika Panda robot. Because the robot is not equipped with a true force sensor, it estimates the force on its manipulator internally. However, this estimation is noisy and biased, and required post-processing before use. By applying low-pass filters and fitting smooth functions over the raw data, most of the noise could be eliminated, but it remains a weak spot in the method.

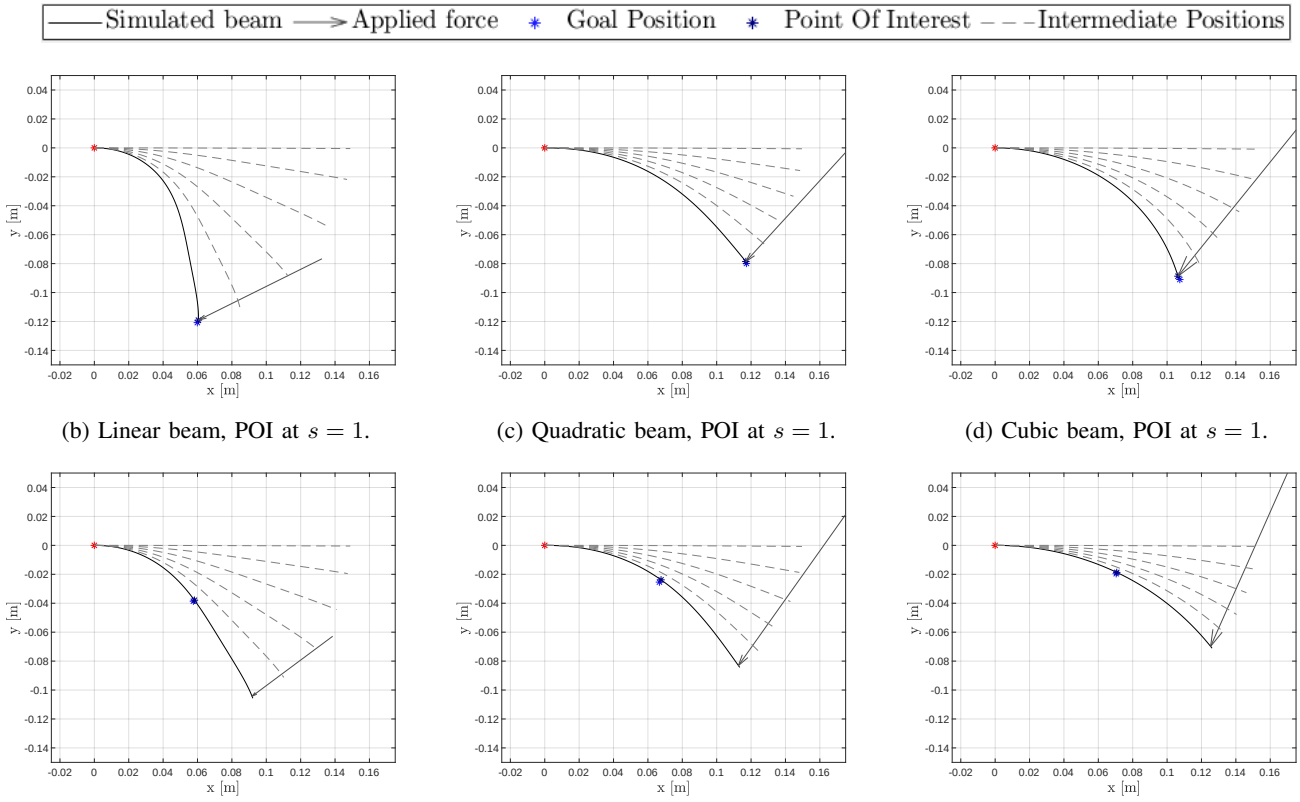
Secondly, the method is sensitive to inaccuracies when estimating the stiffness near the tip of the beam. This is because an applied force at the tip creates the largest moment near the base of the beam and the smallest moment at the tip of the beam. Thus, when the tip of the beam seems to deform even slightly due to inaccuracies in the tag detection, when in reality it does not deform, the algorithm infers that the stiffness of the beam near the tip must be very low: after all, even the small moment at the tip is able create a local deformation. While this sensitivity cannot be eliminated, it can be reduced by using higher definition video data or more accurately placed detection tags.

Additionally, a more thorough validation set could have been gathered. Due to practical restrictions, the data obtained in the validation experiments in which the force was applied at various locations was not as convincing as it could have been. When applying a force to a point in the middle of the beam, the manipulator would occlude one or more of the detection tags, rendering the video data unusable. Thus, an L-shaped tool had to be held by the gripper of the manipulator in order to apply the force to the middle of the beam, but the gripper could not grasp with enough force to apply the required force onto the beam.

Lastly, the manipulation tasks were limited to a simulated environment due to the restrictive capabilities of the Franka Emika Panda. Since this robot is only able to estimate the reactive force at the end-effector, and is not able to exert a specific force on command, it was deemed to not be fit for manipulation tasks.

C. Future Plans

This work can be extended in multiple ways. Firstly, one could remove one or more of the assumptions made regarding the dynamics of the system. For instance, it would be interesting to research the possibility for a simultaneous estimation of the stiffness function and the density function. This can be done by removing the assumption of zero-gravity, and standing the beam upright during experiments. Additionally, the clamping of the beam could be replaced by a true second manipulator, to create a more life-like experiment environment. Doing so would open up possibilities for perturbation techniques, and possibly improve the accuracy and speed of the method. Lastly, due to restrictions regarding the available robotic manipulator, it was not possible to verify the accuracy of the manipulation task on a real-life experiment setting. It would be valuable to see whether the proposed methods are viable in a real-life setting.



(b) Linear beam, POI at $s = 1$. (c) Quadratic beam, POI at $s = 1$. (d) Cubic beam, POI at $s = 1$.

(e) Linear beam, POI at $s = 0.5$. (f) Quadratic beam, POI at $s = 0.5$. (g) Cubic beam, POI at $s = 0.5$.

Fig. 8: Visualisations of the manipulation tasks that were performed in simulation. For each beam, it can be seen that the point of interest is succesfully placed at the desired position.

VII. Conclusion

In this paper, a method has been presented in which the unknown deformation parameters, specifically the stiffness distribution, of a deformable beam are learned using a single experiment. Using a training-validation data split, we verified that the obtained stiffness distribution is an accurate estimation of the true stiffness distribution of the experiment beam. Subsequently, the learned deformation model is used to prove that the shape of the soft object can now be controlled in simulation. Future work includes (i) expanding the linear regression towards estimating both the stiffness function as well as the density function of the deformable beam, (ii) performing experiments using a true bimanual manipulator, and (iii) performing control tasks in a real-life experiment setting.

Acknowledgements

References

[1] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 258–265.

[2] A. Eguchi, *Bringing Robotics in Classrooms*. Cham: Springer International Publishing, 2017, pp. 3–31. [Online]. Available: https://doi.org/10.1007/978-3-319-57786-9_1

[3] J. Billingsley, A. Visala, and M. Dunn, "Robotics in agriculture and forestry," 2008.

[4] M. Kawollek and T. Rath, "Machine vision for three-dimensional modelling of gerbera jamesonii for automated robotic harvesting," in *International Conference on Sustainable Greenhouse Systems-Greensys2004 691*, 2004, pp. 757–764.

[5] Q. Zhang, M. Karkee, and A. Tabb, "The use of agricultural robots in orchard management," *arXiv preprint arXiv:1907.13114*, 2019.

[6] Y. Bai, W. Yu, and C. Liu, "Dexterous manipulation of cloth," *Computer Graphics Forum*, vol. 35, pp. 523–532, 05 2016.

[7] A. Petit, V. Lippiello, G. A. Fontanelli, and B. Siciliano, "Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot," *Robotics and Autonomous Systems*, vol. 88, pp. 187–201, 2017.

[8] H. Lin, F. Guo, F. Wang, and Y.-B. Jia, "Picking up a soft 3d object by "feeling" the grip," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1361–1384, 2015. [Online]. Available: https://doi.org/10.1177/0278364914564232

[9] A. Jordt and R. Koch, "Fast tracking of deformable objects in depth and colour video," 08 2011.

[10] L. Zaidi, J. A. Corrales, B. C. Bouzgarrou, Y. Mezouar, and L. Sabourin, "Model-based strategy for grasping 3d deformable objects using a multi-fingered robotic hand," *Robotics and Autonomous Systems*, vol. 95, pp. 196 – 206, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889016308089

[11] Z. Hu, T. Han, P. Sun, J. Pan, and D. Manocha, "3-d deformable object manipulation using deep neural networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4255–4261, 2019.

[12] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics." in *NIPS*, vol. 27. Citeseer, 2014, pp. 1071–1079.

[13] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, 2018, pp. 39–45.

[14] P. Hyatt, D. Wingate, and M. D. Killpack, "Model-based control

of soft actuators using learned non-linear discrete-time models," *Frontiers in Robotics and AI*, vol. 6, p. 22, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2019.00022

[15] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 823–834, 2015.

[16] G. Tonietti and A. Bicchi, "Adaptive simultaneous position and stiffness control for a soft robot arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2002, pp. 1992–1997 vol.2.

[17] H. Wang, B. Yang, Y. Liu, W. Chen, X. Liang, and R. Pfeifer, "Visual servoing of soft robot manipulator in constrained environments with an adaptive controller," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 41–50, 2017.

[18] M. Trumić, C. Della Santina, K. Jovanović, and A. Fagiolini, "Adaptive control of soft robots based on an enhanced 3d augmented rigid robot matching," *IEEE Control Systems Letters*, 2020.

[19] C. D. Santina and D. Rus, "Control oriented modeling of soft robots: The polynomial curvature case," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 290–298, April 2020.

[20] C. Della Santina, "The soft inverted pendulum with affine curvature," 2020.

[21] C. Della Santina and D. Rus, "Control oriented modeling of soft robots: the polynomial curvature case," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 290–298, 2019.

[22] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.