Robotic Manipulation for Identification of Flexible Objects

T. M. Caldwell and D. Coleman and N. Correll*

Department of Computer Science, University of Colorado,
1111 Engineering Dr, Boulder, CO 80309, USA
{caldwelt,david.t.coleman,nikolaus.correll}@colorado.edu

Abstract. This paper provides preliminary insight into stiffness profile identification of a complex flexible object by robotic manipulation. The object is in the shape of the letter 'Y', chosen to resemble a living plant. The object is approximately modelled as a spring mass system. The robot manipulates the object with one or two arms grasped at the ends of the 'Y', and makes visual measurements which locates the object's position in space. Identification results from an optimization approach are compared for both one and two arm manipulation and sensing with and without vision. The results are not consistent with the expected physical object's properties due to a failure to observe the motion dependence between the object's connected segments. The result provides insight into the problem of assessing the minimal information needed to identify the stiffness of a flexible object, an issue of importance to automated approaches.

1 Introduction

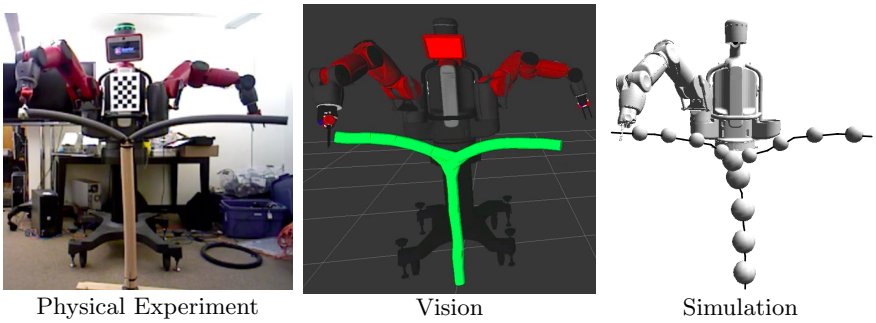

Fig. 1. The physical experiment, vision capture (in green), and simulation of baxter manipulating a flexible object.

* This work was supported by a NASA Early Career Faculty fellowship NNX12AQ47GS02. We are grateful for this support.

The goal of this work is to use a robotic arm and an external vision system to identify the behavior of the flexible object shown in Figure 1. This is an important step toward manipulation of flexible objects such as living plants, rubber tubes, and clothes [22, 19, 2, 6].

There are many methods to model and simulate flexible objects [11, 12]. A common approach is to model the object as a lattice or collection of links of masses and springs [20, 21, 11]. This approach has been used to simulate linear object like strings, hair, and electrical cables for which the model is a series of masses linked together with springs.

We also model the flexible object as a spring mass system. In [3] we modeled a rubber tyre in such a way, and identified the spring constants for its uniform stiffness with a novel identification method. In this paper, we explore a more complex structure in the form of a foam 'Y' made of tubes with differing stiffnesses. We observe that not only must we carefully plan where to manipulate an object in order to sufficiently excite all of its degrees of freedom for measurement—in this case at the end of each Y—but that this is insufficient to accurately identify its stiffness profile, even with additional measurements made by an external vision system. The failure in identification is due to the motion dependence between the object's segments.

Two types of sensing are investigated. The first is the joint angles and torques of Rethink Robotic's Baxter robot's arms. The second is an external vision system that produces a point cloud of the object and through filtering and fitting, can locate specific points on the object. We note that the vision cannot make any torque or force measurements and as such can not directly measure stiffness properties.

We model the object with the same underlying mechanics as the robot arm—i.e., as a collection of rigid bodies connected at joints by springs—allowing us to utilize the vast theory of rigid body mechanics [15]. Also, this enables planning and control to be done in the combined arm and object configuration space instead of only the end effector space or object space. We then use an optimal control approach in [3] for calculating model properties that best match the behavior of the flexible object. The physical experiment, the vision capture, and the model can be viewed in Figure 1.

As in [3], we use variational integrators to simulate the robot and object. Variational integrators can be used to describe discrete-time equations of motion of a mechanical system. They are designed from the least action principle and have good properties that agree with known physical phenomenon like stable energy behavior [16]. All simulations were implemented in `trep` [8, 9], which is a tool to simulate articulated rigid bodies using midpoint variational integrators.

1.1 Organization of this paper

This paper is organized as follows: Section 2 sets up the experiment with the robot and flexible object. In Section 3, the flexible object is modeled as a connection of springs and masses. This section also reviews variational integrators. Section 4 discusses the visual perception system and the techniques to filter and

model fit the measured data. Section 5 reviews the identification algorithm from [3]. Section 6 conducts the identification on the flexible object and compares identification with and without vision, as well as discusses the results..

2 Example Experiment Setup

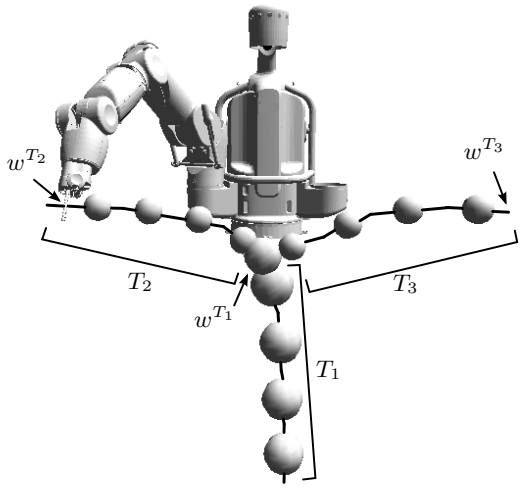

Fig. 2. Model of Baxter manipulating a flexible object. The object is composed of 3 tubes, each approximated with four rigid links. The spheres illustrate the location of the masses and their relative values. The tube segments are labelled T_1, T_2, and T_3, and the points at the ends of the tube are w^{T_1}, w^{T_2}. and w^{T_3}.

The goal of the example experiment is to identify stiffness properties of a flexible object. The flexible object has the shape of the letter 'Y'. It was chosen to resemble the basic geometry of a living plant. The base, or 'trunk' is attached to the ground and is labelled T_1. We investigate two scenarios. In the first, the robot has the point w^{T_2} of branch T_2 grasped, and the branch T_3 is free (refer to Figure 2). In the second scenario, the robot has both points w^{T_2} and w^{T_3} grasped using both arms. The goal is to manipulate the grasped branch so that the movement of the uncontrolled free branch can be predicted.

To improve prediction, we conduct the parameter identification optimization algorithm in [3] in order to identify the model's stiffness properties. The identification is made with physical contact data—i.e. joint angle and torques from one or two arms—and a vision system to capture the motion of the full flexible object. The identification process is the same as in [3] once the vision data has been fitted to the model.

The flexible object is foam tubing with differing widths for each of the T_1, T_2, and T_3 sections. The three tubes are connected with a 'Y' PVC joint and glue. The trunk tube has length 0.813m and mass 0.064Kg. It is the widest tube with radius 0.032m. The grasped tube has length 0.610m and mass 0.015Kg and is the second widest tube with radius 0.025m. The free tube has length 0.737m and mass 0.016Kg. It is the thinnest with radius 0.019m.

We use Rethink Robotics' Baxter robot [5] to both manipulate and measure the object. Baxter's arms each have 7 degrees of freedom. The arms are designed for compliance by means of series elastic actuators in each joint that allows for force sensing and control. Baxter publishes the joint angles and torques at 100Hz. A picture of Baxter manipulating the flexible object is in Figure 1.

3 Model and Simulation

We model the flexible object as a spring mass system by approximating the T_1, T_2, and T_3 segments each with four rigid links of uniform lengths and masses, connected by joints with torsional springs—see Figure 2. Each joint is 3-dimensional allowing for bending and twisting motions of the flexible object. In total, the flexible object is 36 dimensional. The goal of the identification, Section 5, is to identify the torsional springs' spring constants. We label these parameters as ρ.

Due to Baxter's 7 degrees of freedom arms, the system of Baxter grasping one end of the flexible object (neglecting the other arm) has a total number of 43 configuration variables. When grasping both ends of the object, (using both arms) the model has a total number of 50 configuration variables. The dimensions, inertia, and other information concerning Baxter's arm can be obtained at `https://github.com/RethinkRobotics`.

3.1 Simulation

The model for both Baxter's arm and the flexible object are a series of rigid links connected by rotational joints. As such, the dynamics of both the manipulator and the object can be handled together. We use *variational integrators* to simulate the system dynamics. Variational integrators are a discrete-time representation of the equations of motion of a mechanical system. They are designed from the least action principle and have good properties that agree with known physical phenomenon like stable energy behavior [16].

Simulations are for a finite time interval $[0, t_f]$ with discrete times $t_0, t_1, \ldots, t_{k_f}$, where $t_0 = 0$, $t_{k_f} = t_f$ and $k_f + 1$ is the total number of discrete times in the interval. The simulation—i.e. the solving of the system dynamics—will result in a state $x_k := x(t_k)$ for each k. For variational integrators, the state is composed of the configuration, labelled q_k for time t_k, as well as a term labelled p_k, also for time t_k. For systems without external forcing, p_k is the conserved momentum. For the purposes of this paper, it can simply be thought of as analogous to the discrete velocity, which is often paired with q_k to make up the state. The state is $x_k := [q_k, p_k]^T$.

The literature on variational integrators [13] provides a one-step mapping to update the state at the previous time x_k to the next time x_{k+1}. We provide a short high-level review of variational integrators. We write the one-step mapping which constitutes the systems equations of motion as

$$x_{k+1} = f(x_k, \rho, t_k). \tag{1}$$

Here, f explicitly depends on the previous state, time, and the parameters which we wish to identify. While we write the equations of motion as an explicit equation, the equations are in fact implicit and rely on root solving to update the state.

The equations encapsulate the system's Lagrangian, any external forcing, holonomic constraints, as well as a choice of quadrature for approximating integrals. The function f can be linearized. We write

$$A_k = \frac{\partial}{\partial x_k} f(x_k, \rho, t_k) \text{ and } B_k = \frac{\partial}{\partial \rho} f(x_k, \rho, t_k). \tag{2}$$

The equations to calculate the linearization with respect to the state and parameters can be found in [3]. They are needed for calculating the gradient for parameter identification as part of an iterative optimization.

3.2 Simulation of Example

We use variational integrators to simulate Baxter manipulating the flexible object through the simulation tool `trep` [8]. The tool simulates articulated rigid bodies using midpoint variational integrators. It additionally provides partial derivative calculations that we need for the system linearization, Eq.(2).

The system of Baxter manipulating the flexible object with a single arm has a 43 dimensional configuration. Therefore, the system's state, $x_k = [q_k, p_k]^T$, is 86 dimensional. For the system of Baxter manipulating with both arms, the configuration is 50 dimensional and the state is 100 dimensional. The discrete dynamics f, Eq.(1), is given by the discrete system Lagrangian, discrete external forcing, and holonomoic constraints—see [3, 13]. The system Lagrangian is specified by the kinetic and potential energies of Baxter and the object. External forces enter the system through the torques applied by the motors at each of Baxter's joints. Additionally, holonomic constraints are needed to ensure that Baxter's end effectors remain in contact with the object. We chose a time step of 0.01 seconds, which matches the broadcast frequency of Baxter.

Nominally, the simulation will perfectly agree with Baxter's measured joint torque and angles for a given experiment. However, due to model and sensor disturbances, which are always an issue for real systems, this will not be the case. Furthermore, since the system is unstable—i.e. small disturbances can result in large changes to trajectory—directly feeding the measured torques into the model will not result in a meaningful simulation. Therefore, the measured joint torques, labelled $\overline{F}$, and measured joint angles, labelled $\overline{b}$, must be filtered through a feedback loop. We use a simple proportional control law with gain K:

$$F_k = \overline{F}_k - K_k(b_k - \overline{b}_k),$$

where F and b are the simulated joint torques and angles for the filtered control input. When K is large, the effect the parameters have on the simulation is dominated by the control and as such, the system cannot be identified. However, if K is too small, the system will remain unstable and not track the measured trajectory well enough to be meaningful. Correctly choosing K for the purposes of parameter identification of unstable systems is left for future work. For this paper, we chose K from a finite horizon LQR which results in an optimal feedback gain from the model linearized around b_{meas} and a quadratic cost functional (see [1] for LQR). The tradeoff between tracking the joint angles or joint torques is directly represented in the quadratic cost for specifying the size of K.

4 Vision

With the goal to improve parameter identification, the flexible object is visually tracked using an out of the box depth sensor, the Asus Xtion Pro Live. The depth sensor captures a point cloud, G, of the experiment. Processing G consists of three steps: filtering, segmentation, and graph creation.

A filtering component within the motion planning framework MoveIt! [4] performs the first step of self-filtering. It removes points detected on the robots body and arms. This is accomplished using the realtime joint states and calculated coordinate transforms to determine the robots configuration within the point cloud. A small amount of padding is included in the filtering to account for calibration error. The Point Cloud Library (PCL) [18] provides the second level of filtering, removing points detected behind the robot and on the floor. Finally, a statistical outlier removal filter removes remaining noise and measurement errors.

The segmentation of the filtered point cloud $G_{filtered}$ is accomplished using a custom algorithm built on top of PCL. This step converts the T_1, T_2, and T_3 sections (refer to Figure 2) of the object into segmented components that can later be turned into a graph, as shown in Figure 3. The lowest point in the point cloud (aligned with gravity) seeds the algorithm. The k_1 nearest neighbors to this point is then chosen using a Kd Tree to represent a segment s of the plant model. The centroid of s is calculated and a second k_1 nearest neighbors search finds a centered segment $s_{centered}$ at the base of the plant.

Assuming the number of points in $s_{centered}$ are above a minimum threshold (to remove noise), the 3d centroid of $s_{centered}$ is added to a processed graph $G_{processed}$ and the points in $s_{centered}$ are removed from $G_{filtered}$. The next nearest neighbor to $s_{centered}$ is chosen as the new segment starting point and the algorithm repeats. Occasionally, there are insufficient nearest neighbors in a segment if, for example, the algorithm has reached the end of a plant branch. In this case, a random point is chosen in the remaining point cloud $G_{filtered}$ to continue the search, until no further points remain.

The final processing step takes the disconnected points in $G_{processed}$ and performs one final series of nearest neighbor searches to connect the nodes together to represent a flexible object modeled as a series of connected rigid bodies. These

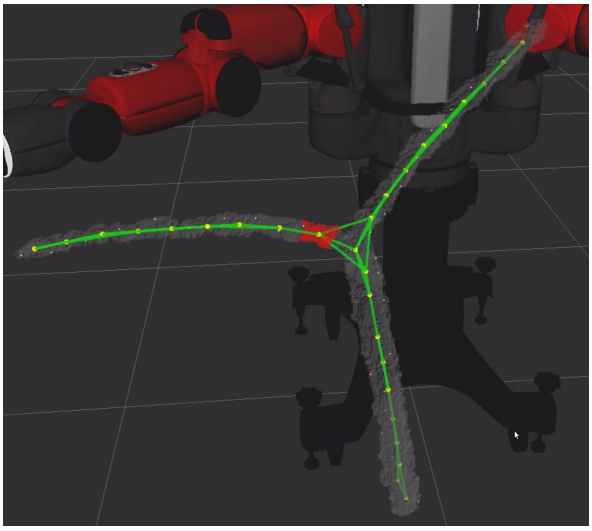

Fig. 3. Vision of flexible object.

connected rigid bodies give a surprisingly accurate three dimensional reconstruction of a flexible object in soft-realtime, processing new point clouds at a rate averaging 3 hz.

4.1 Fitting to Model

The processed points in $G_{processed}$ need to be fitted to the static model of the plant to be useful. The model is a discretization of the physical object where the flexible object's configuration specifies the location of the discretization points. Label the objects configuration as q_o.

The fitting is a calculation on q_o and is accomplished as follows: Let $G_{model}(q_o)$ be a graph specified for configuration q_o with the discrete points as its vertices and adjacent points in the model as its edges. Any two adjacent points in $G_{model}(q_o)$ can be connected by a line segment in space. Let $L_{model}(q_o)$ be the collection of these line segments. Further, let $d(p, \ell)$ be the shortest Euclidean distance between the point $p \in G_{processed}$ and line segment $\ell \in L_{model}(q_o)$. Define $d(p, L_{model}(q_o)) := \min_{\ell \in L_{model}(q_o)} d(p, \ell)$ as the least distance between p and any line segment in $L_{model}(q_o)$. This can be done for each $p \in G_{processed}$. The fitting is given by the q_o for which the points in $G_{processed}$ are nearest the line segments $L_{model}(q_o)$—i.e. by the optimization program

$$\arg\min_{q_0} \sum_{p \in G_{processed}} d(p, L_{model}(q_o)). \tag{3}$$

4.2 Vision Tracking Concerns

A number of tuned parameters make the vision filtering and fitting algorithms sensitive to object size, the distance between the camera and object, and variability of the object's thicknesses. However, it works well for our experimental goals.

One major shortcoming of the vision tracking pipeline developed for this experiment is occasional loss of data due to buffering issues and self occlusion. Because of the computational complexity of our flexible object manipulation pipeline, the experiment was run on 3 distributed commodity PCs using ROS [17]. A common issue is clock time synchronization between the three PCs, and ROS messages being dropped due to full buffers. This causes the robot transforms to be published with old time stamps and the robot self-filtering of the point clouds to stall, ultimately resulting in choppiness in the visual tracking of the plant model. This is an area of continued investigation and improvement.

5 Identification

The goal of the identification is to calculate the system model parameters that best agree with the physical behavior of the flexible object. For the example in the paper, the parameters we wish to identify are the spring constants associated with the flexible object spring mass model.

Each joint of the flexible object is 3 dimensional; each can rotate around each axis. As seen in figure 4, each joint frame has the Z-axis aligned with the link. Therefore, a bend in the tube at a joint is realized by a rotation about the X- and Y-axes and a twist in the tube is a rotation about the Z-axis. The object's configuration specifies the amount each frame is rotated. Because the foam is uniformly distributed for each tube, we assume that the spring constants associated with bending—i.e. rotations about the X and Y axes—are the same for a single tube.

Label the torsional spring constant about the X-axis (alternatively Y or Z) for the i^{th} tube as $\kappa_{T_i,X}$ (alt, $\kappa_{T_i,Y}$ or $\kappa_{T_i,Z}$). There are 6 total parameters $\rho = [\rho_1, \ldots, \rho_6]$ in our model, where

$$\begin{aligned} \rho_1 &= \kappa_{T_1,X} = \kappa_{T_1,Y}, \\ \rho_2 &= \kappa_{T_1,Z}, \\ \rho_3 &= \kappa_{T_2,X} = \kappa_{T_2,Y}, \\ \rho_4 &= \kappa_{T_2,Z}, \\ \rho_5 &= \kappa_{T_3,X} = \kappa_{T_3,Y}, \text{ and} \\ \rho_6 &= \kappa_{T_3,Z}. \end{aligned} \tag{4}$$

The goal of Section 6 is to identify ρ by calculating its corresponding simulation that best matches measured data. This parameter optimization is presented next.

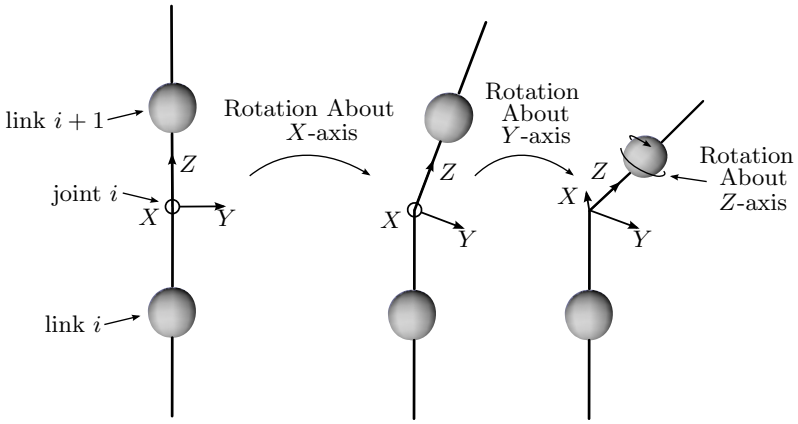

Fig. 4. Illustration of joint i connecting links i and $i+1$ of the flexible object. The joint is a rotation around the X, Y, and Z axes.

5.1 Optimal Parameter Identification

The goal of parameter optimization is to calculate the model parameters ρ that minimize a cost functional. The cost functional is the integral of a running cost $\ell_d(x_k, \rho)$ plus a terminal cost $m(x_{k_f}, \rho)$:

$$\min_{\rho} \left[J_d(\rho) := \sum_{k=1}^{k_f} \ell_d(x_k, \rho) + m_d(x_{k_f}, \rho) \right]$$

constrained to the dynamics, $x_{k+1} = f(x_k, \rho, t_k)$. Since this is a nonlinear optimal controls problem, we turn to iterative methods like steepest descent to calculate a local minima. In order to apply steepest descent, we must have access to the gradient of the cost, which is given in the following Lemma from [3].

Lemma 1. *Suppose $f(x_k, \rho, t_k)$ is $\mathcal{C}^2$ with respect to x_k and ρ. Let A_k and B_k form the linearization of f, Eq.(2), and assume f_k exists. Then,*

$$\nabla J_d(\rho) = \sum_{k=1}^{k_f} \lambda_k B_{k-1} + \frac{\partial}{\partial \rho} \ell_d(x_k, \rho) + \frac{\partial}{\partial \rho} m_d(x_{k_f}, \rho) \tag{5}$$

where λ_k is the solution to the backward one-step mapping

$$\lambda_k = \lambda_{k+1} A_k + \frac{\partial}{\partial x_k} \ell_d(x_k, \rho) \tag{6}$$

starting from $\lambda_{k_f} = \frac{\partial}{\partial x_{k_f}} \ell(x_{k_f}, \rho) + \frac{\partial}{\partial x_{k_f}} m_d(x_{k_f}, \rho)$.

It is worth noting that f_k is not guaranteed to exist, but its existence can be checked using the Implicit Function Theorem. In [7] a couple of scenarios are shown for which such singularities occur. Also, [14] reports the gradient and Hessian for optimal parameter identification in continuous time. The steepest descent direction is $-\nabla J_d(\rho)$ and the steepest descent algorithm can be applied [10].

6 Identification for Example

The parameters to be identified are the spring constants of the flexible object given by the six dimensional ρ. The identification calculates the value ρ with simulation that best matches the measured data, at least locally. We do the matching for two sets of measured data and for two scenarios—i.e. for a total of four experiments. The first measurement set is just Baxter's arm joint torque and joint angle measurements while the second also includes vision (see Section 4). The first scenario is Baxter manipulating one end of the object with a single arm and the second scenario is Baxter grasping both ends using both arms.

The identification is made by minimizing the error between simulated motion for a given ρ and measured motion at one to three points on the object, depending on the available measurements. The three points are the points at the end of the three tubes, labelled w^{T_1}, w^{T_2}, and w^{T_3} (refer to Figure 2). The simulated points are labelled $w_k^{T_1}(\rho)$, $w_k^{T_2}(\rho)$, and $w_k^{T_3}(\rho)$ for parameters ρ at time t_k. The measured points at time t_k are labelled $\overline{w}_k^{T_1}$, $\overline{w}_k^{T_2}$, and $\overline{w}_k^{T_3}$. The point w^{T_1} can only be measured with vision, while the points w^{T_2} and w^{T_3} are measured by the robotic arm when grasped, otherwise they are measured by vision.

As discussed in Section 4.2, vision measurements arrive at irregular intervals and sometimes of poor quality due to missing data. The following process removes bad vision data and aligns the good data with the timing of the simulation: From Section 4, each frame of vision data is processed resulting in points $G_{processed}$ and fitted to the model with optimal fit of $q_o^\star$. Recall $q_o^\star$ is the object's configuration that best fits the data and is calculated from the program Eq.(3). The frame occurs at a time s and so we label that frame's fit as $q_o^\star(s)$. Furthermore, the quality of the fit is quantified by the value of $d(p, L_{model}(q_o^\star(s)))$, where lesser values correspond to better fits. As such, 'good' data is the configurations $q_o^\star(s)$ where $\sum_{p \in G_{processed}} d(p, L_{model}(q_o^\star(s))) < d_{max}$, a user specified tolerance. Data that does not meet this requirement is discarded. In order to align the data timing with the simulation, we first interpolate in time over the remaining data using a cubic spline and label the result $q_{o,interp}^\star$. Second, we calculate the simulation times t_k that are nearest the times s of the remaining data. Define $\sigma = \{\sigma_1, \ldots, \sigma_{k_f}\}$ as $\sigma_k = 1$ if t_k is the simulation time nearest a vision frame time s. In the identification, the cost function depends on the vision data $q_{o,interp}^\star(t_k)$ for which $\sigma_k = 1$, where the points $\overline{w}_k^{T_i}$, $i = 1, 2, 3$, are calculated using forward kinematics.

Aside: Since we have access to $q^{\star}_{o,interp}(t_k)$, we could alternatively choose to minimize the error in the simulated and measured configurations instead of at specific points. However, the position of any point on the object is not uniquely specified by a single configuration. In fact, since the vision system cannot measure a twist in a single tube, every configuration specifying the rotation around its local Z-axis (see Figure 4) is arbitrarily set to 0. This results in a single measured object configuration, but it is unlikely that this choice results in the same configuration as the one simulated, even if the vision perfectly measures the position of all locations on the object.

The identification locally minimizes a cost function J_d given by running cost $\ell_d(q_k, \rho)$ and the terminal cost $m_d(q_{k_f}, \rho)$. Both depend on the error between simulated and measured. Label the errors for each of the three points as:

$$\epsilon_k^{T_1} = (w_k^{T_1}(\rho) - \overline{w}_k^{T_1}),\ \epsilon_k^{T_2} := w_k^{T_2}(\rho) - \overline{w}_k^{T_2},\ \text{and}\ \epsilon_k^{T_3} = (w_k^{T_3}(\rho) - \overline{w}_k^{T_3}).$$

For the measurements that depend on vision, their corresponding error terms will be multiplied by σ_k in the running cost. The identification then locally minimizes J_d using the approach in [3] to calculate the locally optimal parameters $\rho^{\star}$ from an initial guess of $\rho = [3, 3, 3, 3, 3, 3]^T$ and inequality constraint $\rho_i \geq 0$—see Section 5.

The results follow:

6.1 Single Arm and No Vision

Without vision, the only measurements are Baxter's arm joint torques and angles. Baxter is only in contact with the object at the point w^{T_2}, and as such can only measure the flexible object's motion at that single point. This measurement is $\overline{w}_k^{T_2}$. The running cost, ℓ_d, and terminal cost, m_d, in the cost J_d are set as:

$$\ell_d(q_k, \rho) = \frac{1}{2}(\epsilon_k^{T_2})^T \epsilon_k^{T_2} \text{ and } m_d(q_{k_f}, \rho) = \frac{1}{2}(\epsilon_{k_f}^{T_2})^T \epsilon_{k_f}^{T_2}.$$

The locally optimal parameters for the single arm, no vision, case are $\rho^{\star}_{SA,NV} = [23.780, 0.000, 18.357, 11.103, 3.200, 3.059]^T$.

6.2 Single Arm and Vision

With vision, the motion of each point w^{T_1}, w^{T_2}, and w^{T_3} can be measured. Only $\overline{w}_k^{T_2}$ is measured from the robot arm, while $\overline{w}_k^{T_1}$ and $\overline{w}_k^{T_3}$ are measured from vision and are only valid at times t_k where $\sigma_k = 1$.

The running cost is

$$\ell_d(q_k, \rho) = \frac{1}{2}\sigma_k(\epsilon_k^{T_1})^T \epsilon_k^{T_1} + \frac{1}{2}(\epsilon_k^{T_2})^T \epsilon_k^{T_2} + \frac{1}{2}\sigma_k(\epsilon_k^{T_3})^T \epsilon_k^{T_3}.$$

We set the running cost to be $m_d(q_{k_f}, \rho) = \ell_d(q_{k_f}, \rho)$. Executing the identification results in locally optimal parameters $\rho^{\star}_{SA,V} = [22.270, 13.593, 10.735, 8.202, 11.5111, 9.692]^T$ for the single arm with vision measurements scenario.

6.3 Dual Arms and No Vision

With two arms and without vision, only the points w^{T_2} and w^{T_3} are measured. The running and terminal costs are

$$\ell_d(q_k, \rho) = \frac{1}{2}(\epsilon_k^{T_2})^T \epsilon_k^{T_2} + \frac{1}{2}(\epsilon_k^{T_3})^T \epsilon_k^{T_3}$$

and $m_d(q_{k_f}, \rho) = \ell_d(q_{k_f}, \rho)$. The identification results in the locally optimal parameters $\rho^\star_{DA,NV} = [19.559, 4.965, 8.575, 10.374, 0.000, 11.287]^T$.

6.4 Dual Arms and Vision

Finally, with both arms and vision, all points can be measured, but only the points w^{T_2} and w^{T_3} are measured from the arm and w^{T_3} is measured from vision. The running and terminal costs are

$$\ell_d(q_k, \rho) = \frac{1}{2}\sigma_k(\epsilon_k^{T_1})^T \epsilon_k^{T_1} + \frac{1}{2}(\epsilon_k^{T_2})^T \epsilon_k^{T_2} + \frac{1}{2}(\epsilon_k^{T_3})^T \epsilon_k^{T_3}.$$

and $m_d(q_{k_f}, \rho) = \ell_d(q_{k_f}, \rho)$. The optimal parameters are identified as $\rho^\star_{DA,V} = [24.840, 0.000, 12.373, 30.775, 1.406, 24.900]^T$.

6.5 Discussion of Results

The identified model parameters are not consistent with physical behavior. For example, it is unreasonable to expect that a physical tube does not have any stiffness associated with bending or twisting, which is reported by an identified spring constant of zero in all experiments except one. It is worth noting that even the experiment for which the robot manipulates both ends of the object and uses vision to locate the junction point, w^{T_1}, results in a spring constant with value 0.

We expect that this negative result is due to the motion dependence between the object's segments, which was not observable by the object's motion at the measurement locations w^{T_1}, w^{T_2}, w^{T_3}. In other words, we expect that with a perfect model and perfect measurements, the parameters ρ would not uniquely specify the motion of the measurement locations—or, at the least, that the parameters are highly sensitive to disturbances. This explanation is analogous to the observability gramian of linear control analysis being nonsingular.

To illustrate the issue, we look at a simple two spring system. Suppose both springs' spring constants differ and are unknown. One end of one spring is attached to one end of the other string. If the spring system is pulled apart and only the position and forces of the free ends are measured, then the spring constants cannot be identified.

Now, suppose external vision measurements are available and the attachment point can be measured. Then, the displacement of each spring caused by the external forces can be measured and the spring constants can be identified. We

further complicate the system by connecting in series two torsional springs of differing, unknown spring constants. A known torque is applied to the system and the total angular displacement is measured. The goal is to identify the spring constants. However, assuming the vision system not able to measure the angular displacement of either torsional spring, there is once again an identifiability issue even with vision information.

This simple spring example and the identification results of the significantly higher dimensional system in this paper, provide insight into the problem of determining the minimal information needed to assess the stiffness of flexible objects. Solving such a problem would be invaluable to an automated identification routine for identification of other flexible objects with complex shapes or nonuniform stiffnesses. This problem will be addressed in future work.

Even though the identified parameters are not consistent with physical behavior, they are still useful depending on the desired task. For instance, the identified model is valid for planning a manipulation that moves a measurement point to a desired location. This problem is also planned future work.

7 Conclusion

The paper investigates the problem of identifying the stiffness profile of a flexible object shaped like the letter 'Y' through robotic manipulation. The robotic arms and a vision system measure the object's motion. Four experiments are run, once each for each of the following scenarios: manipulation with one or two arms and measurements with or without vision. The identification minimizes the error between the simulated and measured movement of up to three locations on the object depending on the available measurements. The identification results do not match the objects' expected physical properties, which we attribute to a failure to observe the motion dependence of the object's distinct segments. These results, while negative, provide insight into the problem of determining the minimal measurements needed to uniquely identify an object's stiffness.

References

1. B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, INC, 1990.
2. Matthew Bell. *Flexible object manipulation*. PhD thesis, DARTMOUTH COLLEGE Hanover, New Hampshire, 2010.
3. T. M. Caldwell, D. Coleman, and N. Correll. Optimal parameter identification for discrete mechanical systems with application to flexible object manipulation. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.
4. D. Coleman, I. Sucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software. *Journal of Software Engineering in Robotics, Special issue on Best Practice in Robot Software Development*, 2014.
5. E. Guizzo and E. Ackerman. How rethink robotics built its new baxter robot worker. *IEEE Spectrum*, 2011.

6. P Jiménez. Survey on model-based manipulation planning of deformable objects. *Robotics and Computer-Integrated Manufacturing*, 28(2):154–163, 2012.
7. E. Johnson, J. Schultz, and T. D. Murphey. Structured linearization of discrete mechanical systems for analysis and optimal control. *Transactions on Automation Science and Engineering*, 2014. (Accepted).
8. E. R. Johnson and T. D. Murphey. Scalable variational integrators for constrained mechanical systems in generalized coordinates. *IEEE Transactions on Robotics*, 25(6):1249–1261, 2009.
9. E. R. Johnson and T. D. Murphey. Linearizations for mechanical systems in generalized coordinates. In *American Control Conference*, pages 629–633. IEEE, 2010.
10. C. T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, 1999.
11. F. F. Khalil and P. Payeur. Dexterous robotic manipulation of deformable objects with multi-sensory feedback-a review. *Robot Manipulators, Trends and Development, In-Teh (Eds)*, pages 587 – 621, 2010.
12. J. Lang, D. K. Pai, and R. J. Woodham. Acquisition of elastic models for interactive simulation. *The International Journal of Robotics Research*, 21(8):713–733, 2002.
13. J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10(1):357–514, 2001.
14. L. M. Miller and T. D. Murphey. Simultaneous optimal estimation of mode transition times and parameters applied to simple traction models. *IEEE Transactions on Robotics*, 29(6):1496–1503, 2013.
15. R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.
16. D. Pekarek and T. D. Murphey. A backwards error analysis approach for simulation and control of nonsmooth mechanical systems. In *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 6942–6949, 2011.
17. Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.
18. Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE, 2011.
19. Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *Robotics, IEEE Transactions on*, 23(6):1141–1150, 2007.
20. K. S. M. Sahari, C. H. Min, and Y. C. Hou. Dynamic modeling of string for robotics application. In *Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 774–779. IEEE, 2012.
21. H. Wakamatsu and K. Takahashiand S. Hirai. Dynamic modeling of linear object deformation based on differential geometry coordinates. In *International Conference on Robotics and Automation*, pages 1028–1033. IEEE, 2005.
22. Hidefumi Wakamatsu, Eiji Arai, and Shinichi Hirai. Knotting/unknotting manipulation of deformable linear objects. *The International Journal of Robotics Research*, 25(4):371–395, 2006.