Understanding the Impact of Modeling Abstractions on Motion Planning for Deformable Linear Objects

Jimmy Envall, Bernhard Thomaszewski, Stelian Coros

***Abstract*— Robotic manipulation of deformable objects remains challenging due to the high dimensional configuration space and complex dynamics. In this work we demonstrate how the abstraction level used for modeling deformable objects can significantly impact the difficulty of the motion planning problem. We specifically focus on buckling—a nonlinear instability phenomenon that arises in response to compression of slender deformable objects. Using deformable linear objects (DLOs) as a case of study, we show that eliminating resistance to compression in the simulation model while penalizing compressed states in the planning objective increases both robustness and performance. We demonstrate our approach on a set of simulation examples and validate our results through physical robot experiments.**

I. Introduction

Some industries, such as automotive, are increasingly adopting robotics and automation to improve productivity, reliability, and quality. Nevertheless, in other sectors such as garment manufacturing, automation is still in its infancy. A key challenge in this context is the difficulty of manipulation planning for slender objects such as woven fabrics, threads, cords and other deformable materials. For instance, when handling a cord, the robot must be able to reason about the its future state after applying control forces at selected points. This requires a computational model that captures the physics of the cord with sufficient accuracy while keeping computation costs low. Choosing a computational model necessarily involves a compromise between accuracy and complexity. However, as the aphorism "all models are wrong, but some are useful" aptly points out, utility often outweighs perfection [1]. In this work, we show that the model's level of abstraction can be a key factor for successful task execution. We focus on *buckling*—a nonlinear instability phenomenon that arises in response to compression of slender deformable objects. Buckling poses significant numerical challenges for the simulation model and widens the sim-to-real gap. Using deformable linear objects (DLOs) as an example, we demonstrate that a simulation model that abstracts away buckling by eliminating resistance to compression offers significant benefits over nominally more accurate alternatives.

Building on this model, we construct an optimal control framework to manipulate DLOs into desired configurations using multiple grasps (Fig. 1). We illustrate the advantages of our new simulation model with numerical examples and validate its effectiveness for real-world manipulation tasks on a dual-arm robot setup.

The authors are with the Department of Computer Science, ETH, Zurich, Switzerland. jimmy.envall@inf.ethz.ch; bthomasz@inf.ethz.ch; scoros@inf.ethz.ch

This work was supported by a ETH RobotX research grant funded through the ETH Zurich Foundation.

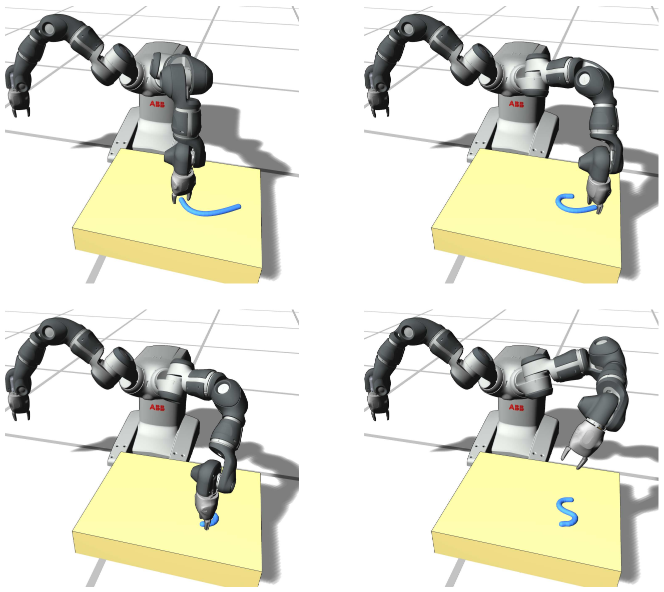


Fig. 1: A dual-arm robot arranging a deformable linear object (DLO) into an "S"-shape in simulation.

Buckling

Buckling is a widely recognized phenomenon [2] that refers to a sudden lateral deflection of a mechanical system subjected to compressive loading. A schematic example is shown in Fig. 2 where three point masses connected with springs are subjected to a compressive force. The post-buckling state of this system is highly sensitive to perturbations and therefore difficult to predict and control.

While buckling occurs with all systems that exhibit slenderness, we focus on the simple case of deformable linear objects confined to a two-dimensional space. As an example, consider the task of moving an initially straight chain, laid out on a table, to a given target location (Fig. 3). A natural strategy is to grasp the end of the chain that is closer to the target and move the chain by pulling. A different, less-than-optimal strategy would be to grasp the opposite end of the chain and push it towards the target. In doing so, the chain ends up in an unstable configuration that easily buckles in different directions, making it all but impossible to reach the desired configuration. Any trajectory that compresses the DLO might cause the object to buckle in an unpredictable manner.

For motion planning applications, buckling is problematic for two main reasons. First, prior to buckling, the system is

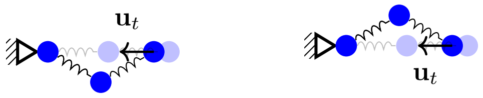


Fig. 2: A mass-spring system with Hookean springs can buckle in different directions when compressed.

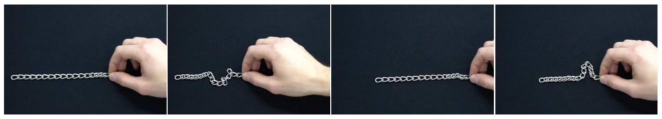


Fig. 3: A steel chain may buckle when subjected to compression. The system is highly sensitive to perturbations, making the post-buckling state difficult to predict.

highly sensitive to perturbations, making the post-buckling state difficult to control. Second, due to the sensitivity around the point of buckling, modeling errors easily lead to a mismatch between simulated and real-world trajectories. The gap between the equilibrium configurations causes the motion planning objective function to become discontinuous, which causes problems when applying gradient-based optimization to the control problem. Numerically, buckling manifests as an indefinite Hessian matrix of the energy [3] and can occur at any point in the optimization process.

II. Related work

A. Manipulation planning

For manipulation planning for deformable objects, researchers often leverage simplifying assumptions regarding the dynamics. For some applications, e.g. cable routing [4], tailored representations can enable the use of existing dynamic programming algorithms. Bretl et al. [5] leverage the quasi static assumption, which enables a sampling based planning strategy over the equilibrium configurations of the object. Yu et al. [6], [7] also leverage the quasi static assumption to learn a mapping that can be used to compute changes in the object's state given the end effector state. Quasi statics can also be combined with position based dynamics into a particularly efficient model that can handle materials of varying stiffness [8]. However, the quasi static framework ignores velocity which makes it difficult to simulate external forces such as friction and drag, which play a critical role when moving objects along a surface.

The quasi static assumption appears to be widely used in the robotics literature, yet there is research that also includes dynamics. For example, Zimmermann et al. [9] use the physically accurate finite element method for manipulation planning with deformable objects while including dynamics. However, they do not discuss buckling, which is more prevalent with thin structures during planar manipulation. In fact, all of [4], [5], [6], [7], [8], [9] focus primarily on applications where friction and damping forces are minimal, such as in-air manipulation with fixed grasps. It is not immediately evident how they might be modified for applications involving planar manipulation with varying contact points.

Machine learning has shown great potential in applications involving uncertainty and external disturbances [10], [11]. However, achieving good performance hinges on having a large amount of data. A practical method for generating data is through simulation which in turn limits the accuracy of the models due to what is known as the notorious sim-to-real gap. In some cases the sim-to-real gap can be efficiently bridged by refining the action based on the residual [12], however, this method applies mainly to tasks that can be safely repeated. Adebola et al. demonstrate the use of imitation learning for autonomous gasket insertion [13], however, collecting human demonstrations is resource intensive and presents scalability issues. Methods based on self-supervision such as that of Lee et al. [14] partially mitigate this problem since data can be collected without human supervision. Our research complements these learning based works by providing a model suitable for use in a traditional optimal control framework.

B. Modeling and simulation

A deformable linear object (DLO) is an object that is significantly longer in one dimension compared to the others and that is flexible. Examples are ropes, hoses and chains. Different models for deformable objects have been researched in great depth in different fields ranging from computational mechanics [15] to computer graphics [16], [17], [18]. In robotics, several different models have proven to be effective, but no single model seems to consistently outperform the others in every situation [19].

Within the computer graphics community, a notable work by Bergou et al. [20] shows a principled approach applicable to ropes, Slinkys and other one dimensional objects. The formulation captures complex mechanical behavior such as perversions arising from twisting. We leverage the bending stiffness component from this work for improved generality and realism in our model.

The buckling phenomenon is well understood, and various methods have been proposed to remedy the associated numerical problems. Duenser et al. [21] use Newton's method for solving the equilibrium state of a steel rod manipulated by a dual-arm robot. When an equilibrium is reached, the Hessian is tested for indefiniteness. If the current equilibrium is not a local minimum then the state is perturbed in a direction that leads towards a local minimum. This is a valid strategy if the selected direction leads to an equilibrium that is useful, but it is hard to know a priori which minimum the selected perturbation will lead to if there are multiple options. For the application targeted by Duenser et al., it is sufficient to find a stable configuration that can be modified throughout the trajectory. This involves maximizing the stability of the equilibrium state using a heuristic.

Indefiniteness arises not only with one-dimensional models with large deformations. With tetrahedral meshes, problems such as element inversion frequently occur, particularly when the simulation involves substantial forces and long time steps [22], [17]. Teran et al. [17] propose to clamp the negative eigenvalues of the block components of the Hessian. This consistently provides a valid descent direction during optimization, and when combined with a modified constitutive model [23], it prevents non-physical artifacts such as element inversion. However, in contrast to element inversion, buckling is not a numerical artifact but a real physical phenomenon resulting from the structural properties of the object. In our application, selectively inhibiting the emergence of certain buckling modes would compromise the accuracy of the model.

Garments and fabric in general are particularly prone to buckling due to the relatively low bending stiffness. Choi et al. presented a remarkable method for realistic simulation of large pieces of fabric [3]. They used a modification of the popular mass-spring model [16] where, similarly as in this work, the springs store only tensile energy and simulate buckling with an additional force component constructed based on fabric-specific behavior. Their method yields realistic-looking cloth behavior and enables simulation with comparatively large time steps. However, even if the simulation captures buckling, the behavior remains sensitive to disturbances, and it is unlikely that the simulated wrinkles perfectly match the wrinkles that would emerge in reality.

Various methods for adjusting an indefinite Hessian to obtain a valid descent direction with Newton's method are documented in the optimization literature [24]. These adjustments are helpful in finding an equilibrium, or a local minimum of the objective function. However, they do not aid in resolving the ambiguity that arises when multiple local minima, and thus equilibrium states, exist near each other.

III. Method

In this section we detail how the simulation model can be built in conjunction with the optimization objective into a formulation that does not suffer from the numerical issues caused by buckling, while still maintaining sufficient accuracy for real-world application. We start with a description of the simulation model in Section III-A, and in Section III-C we detail an optimal control formulation that works in conjunction with the simulation model.

A. Abstracted deformable object model

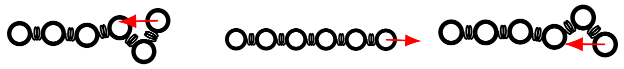
We base our formulation on the popular mass spring model [16]. We write the position of each point mass i as $\mathbf{x}_t^i \in \mathbb{R}^2$. A standard Hookean spring with potential

$$U_h(\mathbf{x}_t^i, \mathbf{x}_t^j) = \frac{1}{2}k(\|\mathbf{x}_t^i - \mathbf{x}_t^j\| - L)^2$$

stores energy both when compressed and extended with respect to the rest length L.

Hookean springs induce indefiniteness when compressed [3]. An indefinite Hessian indicates a saddle point in the energy function, which implies that there are multiple local minima in the forward simulation. This results in ambiguities, as there are multiple possible equilibrium states (Fig. 2). Consequently, the forward simulation becomes sensitive to perturbations; even minor changes, such as slightly different initial guesses can cause the trajectory to shift to a different equilibrium. A control problem that relies on such a forward simulation is discontinuous, which poses problems when

(a)



(b)

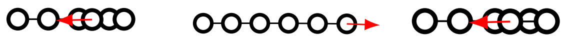


Fig. 4: Repeating the experiment of Fig. 3 with (a) Hookean and (b) unilateral springs in simulation. Hookean springs accurately model the sensitive buckling behavior while unilateral springs do not. The applied force is indicated by the red arrow.

solving the motion planning problem using gradient based optimization methods. As Choi et al. pointed out, the amount of indefiniteness can be arbitrarily large and cannot be fully resolved by using a smaller simulation step size alone [3]. Thus in the first part of the formulation we replace the Hookean springs with unilateral springs whose potential is a piecewise cubic function

$$U_s(\mathbf{x}_t^i, \mathbf{x}_t^j) = k\left[\max\left(\|\mathbf{x}_t^i - \mathbf{x}_t^j\| - L, 0\right)\right]^3.$$

The unilateral springs store energy only when stretched and not when compressed. In practice this means that a spring can be compressed effortlessly, which prevents buckling from happening. The idea is related to tension field theory [25], [26] for membranes and surfaces where the potential for a surface element reaches zero before buckling can happen. This eliminates a large source of indefiniteness in the model and improves convergence in practice due to the lesser need for regularization when solving the forward simulation.

A model that ignores compressive energy in the object (Fig. 4) is inconsistent with common real-world DLOs, such as ropes and hoses. However, by choosing the control strategy appropriately (Section III-F) we can avoid the configurations where the model constitutes a bad approximation of the real system. In Section IV-A we showcase how a model that abstracts away buckling leads to significant improvements in planning performance compared to a physically more accurate model.

B. Motion Planning

We decompose the motion planning problem into two parts; finding a set of contact points and an associated actuation that, when executed, leaves the object in the desired configuration. Once the contact points have been selected (Section III-E), we frame the motion planning problem as a classical optimal control problem (Section III-C) that we solve using gradient based optimization methods.

C. Control problem

The state of the object at time t is denoted by $\mathbf{x}_t \in \mathbb{R}^{2n_p}$ where n_p is the number of point masses. Actuation happens in the horizontal plane by applying control forces $\mathbf{u}_t^i \in \mathbb{R}^2$ where i is the point mass index. The stacked control input for one step is $\mathbf{u}_t \in \mathbb{R}^{2n_p}$. The discrete form of the objective function can then be written as

$$\min_{\mathbf{x}_t, \mathbf{u}_t,\, t \in \{1,\ldots,t_f\}} \quad \mathcal{J}(\mathbf{x}, \mathbf{u}).$$

where $\mathbf{x}$ and $\mathbf{u}$ denote the stacked state and control vectors containing all time steps respectively. We opt for a first order formulation where $\mathbf{x}_t$ contains only the position of the deformable object. The dynamics constraint thus reads $\Delta \mathbf{v}_{t+1} = hf(\mathbf{x}_{t+1}, \mathbf{u}_{t+1})$ where $\Delta \mathbf{v}_{t+1} = \mathbf{v}_{t+1} - \mathbf{v}_t$ and $\mathbf{v}_{t+1} = (\mathbf{x}_{t+1} - \mathbf{x}_t)/h$ for step length h and dynamics function f. We use this to formulate the object state as a function of the control input $\mathbf{x}(\mathbf{u})$. The problem then reads

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathcal{J}(\mathbf{x}(\mathbf{u}), \mathbf{u}) \\ \text{s.t.} \quad & \Delta \mathbf{v} = hf(\mathbf{x}, \mathbf{u}) \\ & \mathbf{x}_{t_0} = \mathbf{x}_0 \end{aligned} \tag{1}$$

where the dependence on t has been omitted for brevity. The boundary condition is the initial state $\mathbf{x}_{t_0} = \mathbf{x}_0$.

In order to apply gradient based optimization methods on Eq. (1) we need to compute the gradient $\nabla_{\mathbf{u}} \mathcal{J}$. This can be done using sensitivity analysis [27], [28]. The total derivative of the objective $\mathcal{J}$ w.r.t to the control input $\mathbf{u}$ is

$$\frac{d\mathcal{J}(\mathbf{x}, \mathbf{u})}{d\mathbf{u}} = \frac{\partial \mathcal{J}}{\partial \mathbf{x}} \frac{d\mathbf{x}}{d\mathbf{u}} + \frac{\partial \mathcal{J}}{\partial \mathbf{u}}. \tag{2}$$

Furthermore, we define the residual of the dynamics $\mathbf{r} = \Delta \mathbf{v} - hf(\mathbf{x}, \mathbf{u})$. Assuming $\mathbf{r} = \mathbf{0}$ everywhere implies that $\frac{d}{d\mathbf{u}} \mathbf{r} = 0$, which in expanded form reads

$$\frac{d\mathbf{r}}{d\mathbf{u}} = \frac{d\mathbf{r}}{d\mathbf{x}} \frac{d\mathbf{x}}{d\mathbf{u}} + \frac{\partial \mathbf{r}}{\partial \mathbf{u}} = \mathbf{0},$$

from where we can solve the sensitivity matrix $\mathbf{S} := \frac{d\mathbf{x}}{d\mathbf{u}}$ as

$$\mathbf{S} = -\left(\frac{d\mathbf{r}}{d\mathbf{x}}\right)^{-1} \frac{\partial \mathbf{r}}{\partial \mathbf{u}},$$

which is then substituted into Eq. (2).

Our implementation is based on Gauss-Newton, for which the approximate Hessian $\mathbf{H}$ can be computed as

$$\mathbf{H} = \mathbf{S}^T \frac{\partial^2 \mathcal{J}}{\partial \mathbf{x}^2} \mathbf{S} + 2\mathbf{S}^T \frac{\partial^2 \mathcal{J}}{\partial \mathbf{x} \mathbf{u}} + \frac{\partial^2 \mathcal{J}}{\partial \mathbf{u}^2}. \tag{3}$$

We refer the reader to, e.g., [28] for a derivation of Eq. (3).

D. Dynamics

We formulate the forward simulation as a standard energy minimization problem. The total spring potential

$$E_s(\mathbf{x}_{t+1}) := \sum_{i \in \{1,\ldots,n_p-1\}} U_s(\mathbf{x}^i_{t+1}, \mathbf{x}^{i+1}_{t+1})$$

is the sum of the potential of the individual springs connecting the n_p point masses (Section III-A), and $E_{\mathbf{u}} = -\mathbf{u}^T \mathbf{x}$ is the pseudo-potential of the control force. We emulate friction using viscous damping with the pseudo-potential $E_v(\mathbf{v}_{t+1}) := (h\mu)/2\mathbf{v}^T_{t+1}\mathbf{v}_{t+1}$. Simulating friction accurately is a complex task, and although some differentiable methods do exist in the literature [27], [29], we found viscous damping to yield satisfactory results in our experiments.

Many DLOs, such as cables, wires and ropes, show some level of resistance to bending. To model this behavior, we include the bending energy in $E_b(\mathbf{x}_{t+1})$. We use the formulation for an isotropic bending response presented by Bergou et al. [20] where the bending modulus is denoted by α. The energy to be minimized for each time step then reads

$$\begin{aligned} \phi(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_{t+1}) = \frac{1}{2} & \Delta \mathbf{v}^T \mathbf{M} \Delta \mathbf{v} \\ & + E_s(\mathbf{x}_{t+1}) + E_{\mathbf{u}}(\mathbf{u}_{t+1}) + E_b(\mathbf{x}_{t+1}) + E_v(\mathbf{v}_{t+1}) \end{aligned}$$

where $\mathbf{M}$ is the diagonal mass matrix. We solve $\min_{\mathbf{x}_{t+1}} \phi$ for each time step until $\|\nabla \phi\| \leq 10^{-12}$ using Newton's method stabilized with line search [24].

E. Contact points

Our control strategy assumes that the contact points have already been selected. Methods for choosing contact points exist in the literature [30], [31], [32]. However, for evaluating our control formulation, a simpler method suffices.

We split the planning horizon into segments of equal length. For each segment we choose one grasping point from a set of potential contact points and set it as "active". By selecting an appropriate number of segments and possible contact points, we can reduce the problem to a manageable set of candidate sequences, allowing us to conduct an exhaustive search.

F. Control Objective

To guide the system towards the goal configuration we use a cost function of the form

$$j_{\text{goal}}(\mathbf{x}) = c_{\text{goal}} \sum_{i \in \mathcal{I}} (\hat{\mathbf{x}}^i_{t_f} - \mathbf{x}^i_{t_f})^T (\hat{\mathbf{x}}^i_{t_f} - \mathbf{x}^i_{t_f})$$

where $\hat{\mathbf{x}}^i$ is the desired position of the i:th point mass at the end of the trajectory and $\mathcal{I} := \{1, ..., n_p\}$ respectively. In addition, we also discourage movement at the end of the trajectory by introducing

$$j_{\text{term}}(\mathbf{x}) = c_{\text{term}} \sum_{i \in \mathcal{I}} \mathbf{v}^{iT}_{t_f} \mathbf{v}^i_{t_f}.$$

The terms j_{goal}, j_{term} could be formulated as constraints. However, in practice, for some contact point sequences it can be difficult to find a control strategy that brings the system to the goal configuration exactly and therefore it is more convenient to impose the goal term as a penalty term rather than writing the problem as a boundary value problem.

The unilateral springs discussed in Section III-A do not resist compression at all. A simulated trajectory that contains compression will not carry over to the real world since physical objects will buckle when compressed. Therefore we want the computed trajectory to contain as little compression as possible. To prevent compression we introduce the following constraint:

$$l^s_t / L - 1 \geq 0 \; \forall \, t \in \mathcal{T},\, s \in \mathcal{S}. \tag{4}$$

for $\mathcal{S} := \{1, ..., n_S\}$, $\mathcal{T} := \{1, ..., t_f - 1\}$ where l_t^s is the current length of spring s at time t. In practice we implement Eq. (4) as a penalty term:

$$j_{\text{comp}}(\mathbf{x}) = c_{\text{comp}} \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}} \left[\min\left(l_s^t / L - 1, 0\right)\right]^3 .$$

A practical issue, especially with longer point mass chains, is that the system might fold over itself. This is undesirable when manipulating in the plane since the chain in reality would collide with itself. As a remedy for large local bends we limit the angle between consecutive spring elements using the following constraint:

$$\mathbf{d}_t^{sT} \mathbf{d}_t^{s+1} \geq \hat{\alpha} \; \forall \, t \in \mathcal{T}, \, s \in \mathcal{L} \tag{5}$$

where $\mathbf{d}_t^s$ is a unit vector in the direction of spring s at time t. The penalty of Eq. (5) reads

$$j_{\text{fold}}(\mathbf{x}) = c_{\text{fold}} \sum_{t \in \mathcal{T}} \sum_{s \in \{1, ..., n_s - 1\}} \left[\min\left(\mathbf{d}_t^{sT} \mathbf{d}_t^{s+1} - \hat{\alpha}, 0\right)\right]^3$$

We set $\hat{\alpha} = 0$ to penalize bends larger than 90° between consecutive spring elements.

Finally we add three regularizers on the control forces, point mass velocities and accelerations respectively:

$$j_{\mathbf{u}}(\mathbf{u}) = c_{\mathbf{u}} \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} \mathbf{u}_i^{tT} \mathbf{u}_i^t,$$

$$j_{\text{vel}}(\mathbf{x}) = c_{\text{vel}} \sum_{t \in \{t_1, ..., t_f\}} \sum_{i \in \mathcal{I}} \Delta \mathbf{x}_i^{tT} \Delta \mathbf{x}_i^t,$$

and

$$j_{\text{acc}}(\mathbf{x}) = c_{\text{acc}} \sum_{t \in \{t_1, ..., t_f\}} \sum_{i \in \mathcal{I}} \Delta \mathbf{v}_i^{tT} \Delta \mathbf{v}_i^t.$$

These terms aid in creating smooth and controlled movements. The final objective is then of the form

$$\begin{aligned} \mathcal{J}(\mathbf{x}, \mathbf{u}) = j_{\text{goal}}(\mathbf{x}) + j_{\text{term}}(\mathbf{x}) + j_{\text{comp}}(\mathbf{x}) \\ + j_{\text{fold}}(\mathbf{x}) + j_{\mathbf{u}}(\mathbf{u}) + j_{\text{vel}}(\mathbf{x}) + j_{\text{acc}}(\mathbf{x}). \end{aligned}$$

IV. Results

We investigate how abstracting away buckling impacts planning performance and how the compression prevention constraint Eq. (4) affects the trajectory by a set of simulated experiments in Section IV-A. In Section IV-B we generate and deploy trajectories on a real robot to validate the model.

A. Numerical study

We begin with a numerical study to illustrate the differences between the two spring types discussed in Section III-A and the impact of the compression prevention term in Section III-F. For the purpose of the study we here choose the contact points manually, while in Section IV-B the contact points are found using the method outlined in Section III-E.

The experiments feature a linear point mass system with eight springs ($n_s = 8$) each with a rest length L of 0.01 m and a bending stiffness of $\alpha = 10^{-9}$. The remaining parameters are listed in Table I.

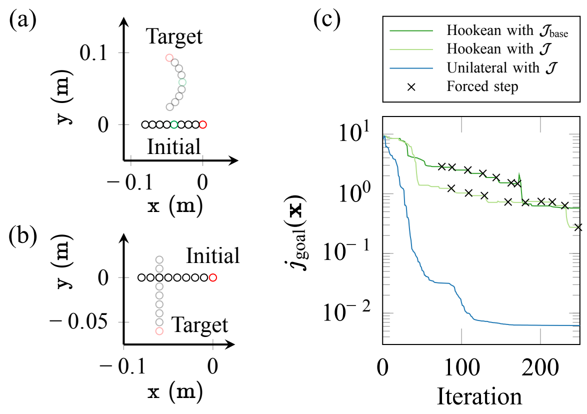


Fig. 5: (a) An arrangement task involving two grasps. (b) A reorientation task involving one grasp. (c) Value of j_{goal} as a function of the iteration number for two different spring types for the arrangement task in (a). The compression prevention constraint does not prevent buckling from occurring when using Hookean springs. Additionally, with Hookean springs the line search fails intermittently due to buckling, and forcing a step in these cases is hardly beneficial.

TABLE I: Parameters used in the experiments.

k	10^3	c_{term}	$0.1/n_P$	c_{vel}	1
h	0.2	c_{comp}	$-10/n_S$	c_{acc}	1
t_f	45	c_{fold}	$-10^5/n_S$	μ	0.5
c_{goal}	$2000/n_P$	$c_{\mathbf{u}}$	$0.1/n_P$		

In the first experiment the system is to be placed in a curved shape above the initial configuration, as shown in Fig. 5a. The control point is initially chosen to be the end point mass marked in red, but changes halfway through the trajectory to the point mass marked in green. We define a new control objective

$$\begin{aligned} \mathcal{J}_{\text{base}}(\mathbf{x}, \mathbf{u}) := j_{\text{goal}}(\mathbf{x}) + j_{\text{term}}(\mathbf{x}) \\ + j_{\mathbf{u}}(\mathbf{u}) + j_{\text{fold}}(\mathbf{x}) + j_{\text{vel}}(\mathbf{x}) + j_{\text{acc}}(\mathbf{x}) \end{aligned}$$

and compare it with $\mathcal{J}$ from Section III-F. The problem is then solved with both unilateral and Hookean springs combined with $\mathcal{J}_{\text{base}}$ and $\mathcal{J}$ for 250 iterations.

Figure 5c shows the value of j_{goal} as a function of the iteration number for both Hookean and unilateral springs. With Hookean springs the optimizer makes little progress, and the value of j_{goal} remains large. In addition, the line search fails intermittently during the optimization. This can happen when the object is compressed and the springs suddenly buckle in a direction that is unfavorable for the objective (see Section I for a discussion on buckling).

A straightforward attempt at alleviating the line search failure would be to force the optimizer to make a step in the search direction whenever the line search fails and accept an increase in the objective value. However, as illustrated in Fig. 5c, with this strategy the final configuration of the point mass system remains far from the target. With unilateral springs, line search does not fail, but instead the compression prevention term j_{comp} must be included to avoid controls

TABLE II: Contact point sequences of the different experiments. Indicated is the spring index of the contact point.

	1st contact	2nd contact	3rd contact
"C"-shape	7	19	1
"R"-shape	7	19	1
"L"-shape	13	7	19
"S"-shape	7	19	1

that cause the springs to compress. A control that causes the object to compress would cause the object to buckle when transferring the trajectory to a real system.

As illustrated in Fig. 5c, the compression prevention constraint hardly aids convergence when using Hookean springs. This happens since the Hookean springs buckle rather than compress in the forward simulation. When the springs buckle, the compressive energy is released, which effectively hides the compression from the control objective. This can happen regardless of the value of the penalty factor c_{comp} since the control objective is evaluated only after unrolling the forward simulation. The unilateral springs do not buckle in the simulation, but instead they compress. However, the compression prevention component of the control problem ultimately leads the optimizer to favor control forces that avoid introducing compression into the system. As the control results in no compression, the object would not buckle in reality either.

The unilateral springs significantly benefit the runtime. For this experiment we measure a 570% increase in runtime when using Hookean springs compared to unilateral springs. The buckling of the Hookean springs is visible as an indefinite Hessian in the forward simulation (Section III-D), and in order to find a valid descent direction the Hessian needs to be regularized. This in turn impedes the convergence of the forward simulation. In addition, at the control level, different step lengths during the line search can cause the forward simulation to buckle in different ways. In situations where the DLO is about to buckle in a way that is unfavorable in terms of the control objective, the optimizer is forced to take small steps, which further increases the runtime.

We study the impact of the compression prevention constraint in a second experiment in which the DLO is to be reoriented as shown in Fig. 5b. The compressive energy stored in the springs throughout a trajectory is $U_{\text{comp}}(\mathbf{x}) := \sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} k/2\left[\min\left(l_s^t - L, 0\right)\right]^2$. With $\mathcal{J}_{\text{base}}$ we obtain $U_{\text{comp}}(\mathbf{x}) = 0.0806$ J, while with $\mathcal{J}$ the corresponding number is only 0.0001 J. The control obtained with $\mathcal{J}_{\text{base}}$ is therefore more likely to cause the DLO to buckle in reality, causing an increase in the sim-to-real gap.

B. Robot Experiments

We validate the control strategy from Section III using an ABB YuMi robot (Fig. 7). To demonstrate the efficacy of the model we run experiments both in an open- and in closed-loop fashion. Some materials, such as steel chains, transfer directly to the real world while more complex materials benefit from a closed-loop solution due to the inevitable mismatch between the simulated material and reality. For all experiments we use 4 potential contact points that are evenly distributed along the DLO. We divide the trajectory into three segments of equal length which allows the contact point to be changed twice throughout the planning horizon. In total this yields 64 different grasp sequence candidates. Actuation happens from the contact point, which is located in the center of the spring, and the control forces are distributed equally to the connecting point masses. We use a fixed computation budget, and terminate the optimization for candidates that have not converged in 500 iterations. From the converged candidates we select the one with the lowest energy. We use a gradient l^2-norm of 10^{-5} as convergence criterion. Our solver for the control problem (Section III-C) is implemented in Julia 1.10 and uses both automatic and symbolic differentiation for computing derivatives [33], [34].

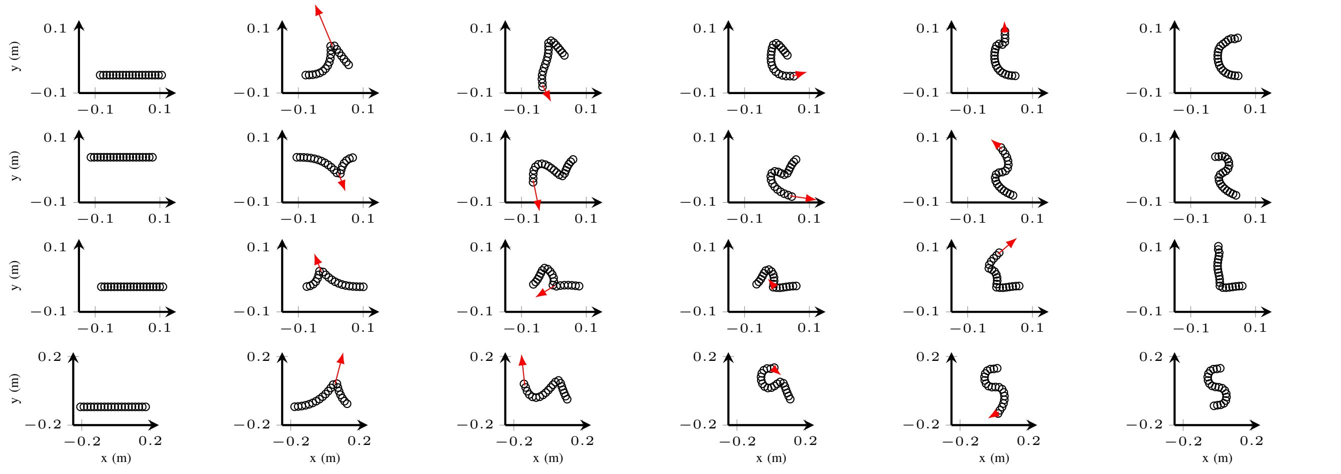
1) Steel chain: We model a steel chain with 20 point masses and 19 springs with $L = 0.01$ m. The physical chain used in the experiments is 20 cm long. The bending stiffness is small and thus we set $\alpha = 0$. The remaining parameters are listed in Table I.

Reaching the convergence criterion requires 92, 40 and 249 iterations for the "C"-, "R"- and "L" letters respectively. The resulting trajectories are shown in Fig. 6, and the sequence of contacts in Table II where the number indicates the spring index used as contact point. Figure 8 shows the progress of the optimization for the different experiments for both unilateral and Hookean springs. The plot shows how the progress stagnates when Hookean springs are used.

The trajectory is transferred to the robot using kinematic trajectory optimization in an open-loop setting as follows: Every manipulation phase is preceded by an alignment phase that aligns the end effector with the active contact point. A manipulation phase is succeeded by a retraction phase in which the end effector is lifted from the manipulation plane. The resulting object shape is shown in Fig. 7.

2) Cotton cord: We model a cotton cord with 20 point masses and 19 springs with $L = 0.02$ m. The cotton cord shows slight resistance to bending and therefore we set $\alpha = 10^{-9}$. The physical cord used in the experiments is 38.0 cm long. The remaining parameters are listed in Table I. The experiment reaches the convergence criterion in 57 iterations. The resulting "S"-shape is shown in Fig. 6. Both the steel chain and the cotton cord trajectories are computed on a workstation equipped with an AMD Ryzen Threadripper 5995WX 64 core CPU where we observe an average runtime of 164 ms for the forward simulation and 1.1 seconds per iteration of the control problem for a single contact sequence.

The closed-loop setting is similar to the open-loop setting, except that the next contact point is estimated from colored markers using an Intel RealSense D455 combined depth and RGB camera. The end effector trajectory is computed so that it minimizes the squared distance to the simulated trajectory, starting from the estimated contact point position. The resulting configuration is shown in Fig. 7.



(a) Initial state. (b) At 1.8 seconds. (c) At 3.6 seconds. (d) At 5.4 seconds. (e) At 7.2 seconds. (f) Final state.

Fig. 6: Snapshots of the control and trajectory for a "C"-shape, a "R"-shape and a "L"-shape using a steel chain and a "S"-shape using a cotton cord. The red arrow indicates the contact point and the force direction.

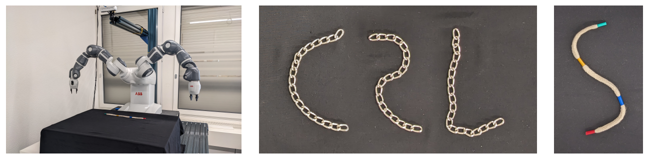


Fig. 7: We validate the generated trajectories on an ABB YuMi robot (left). Letters from the "Computational Robotics Lab" logo (center) formed with a steel chain and a "S"-shape (right) formed with a cotton cord.

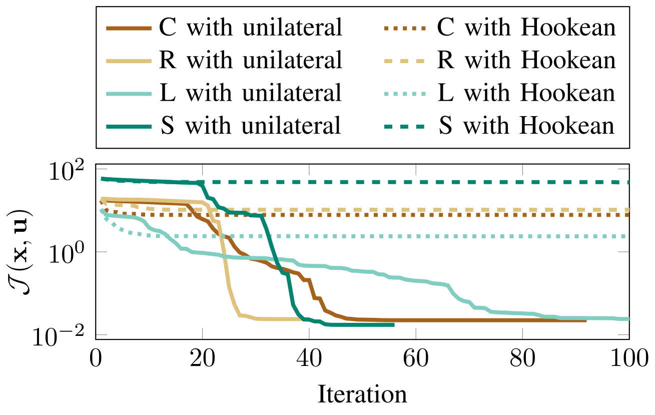


Fig. 8: Objective function value as a function of the iteration number for the experiments and contact sequences listed in Table II with both unilateral and Hookean springs. Hookean springs are preventing the optimizer from converging.

V. Discussion, Limitations and Conclusion

Our results show that abstracting away physical behavior, such as buckling, when simulating deformable objects is advantageous for the motion planning process. By ignoring compression energy in the forward simulation and simultaneously penalizing compression in the control objective, we are able to generate motion plans that transfer to reality. This work focuses on DLOs, but buckling also occurs with objects of higher dimensionality, as discussed in Section II. Extending the formulation to sheets in 3D would form an interesting extension which we defer to future work.

We note that the steel chain used in our real-world experiment is a forgiving choice due to its relatively large weight, which ensures reliable contact with the surface and reduces variability in friction. More complex materials can be manipulated in a closed-loop setting, where sensors compensate for modeling errors that arise from an imperfect model and inconsistent material properties. Additionally, sensing could be combined with differentiable simulation [27] to estimate the drag coefficient and enable replanning on the fly, a direction we leave for future work.

Equation (5) prevents local folds between consecutive spring elements. In practice, this turns out to be sufficient for solving the problems in Section IV without self-intersections, but it only prevents large bends locally. A more general solution would be to simulate collisions and avoid them in the control objective using differentiable collision techniques [35], [29], which we defer to future work.

Our model emulates friction as viscous damping. As the results show, damping yields satisfactory results in some settings. Simulating friction is difficult, however, some differentiable methods do exist in the literature [27], [29], and incorporating these would pose an interesting extension.

A drawback of using gradient based optimization for motion planning is the local nature of the method. However, in practice we encountered few issues with local minima in our experiments, possibly because of the grasp sequence search which hides unfavorable contact sequence choices.

The grasp sequence search is necessary to study the effectiveness of the devised control formulation, but our simple implementation does not scale well with the number of contact points in terms of computational workload. Future research should aim to incorporate contact point selection directly into the control problem, similar to the approach used in integrated Task and Motion Planning (TAMP) [36],

[37]. It has been noted that TAMP involving deformable objects remains largely unexplored [38], and we hope that the concepts presented in this work will be beneficial for future work in this area.

VI. Acknowledgments

We would like to thank Dongho Kang, Valentin N. Hartmann, Miguel Zamora and Yue Li for feedback, ideas and insightful discussions during the course of this work.

References

[1] G. E. Box and N. R. Draper, *Empirical model-building and response surfaces.* John Wiley & Sons, 1987.

[2] E. Riks, "The Application of Newton's Method to the Problem of Elastic Stability," *Journal of Applied Mechanics*, vol. 39, no. 4, pp. 1060–1065, 12 1972.

[3] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth," *ACM Transactions on Graphics*, vol. 21, no. 3, p. 604–611, 2002.

[4] A. Keipour, M. Bandari, and S. Schaal, "Efficient spatial representation and routing of deformable one-dimensional objects for manipulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2022, pp. 211–216.

[5] T. Bretl and Z. McCarthy, "Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 48–68, 2014.

[6] M. Yu, H. Zhong, and X. Li, "Shape control of deformable linear objects with offline and online learning of local linear deformation models," in *2022 International Conference on Robotics and Automation (ICRA).* IEEE, 2022, pp. 1337–1343.

[7] M. Yu, K. Lv, H. Zhong, S. Song, and X. Li, "Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 417–436, 2022.

[8] F. Liu, E. Su, J. Lu, M. Li, and M. C. Yip, "Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 3964–3971, 2023.

[9] S. Zimmermann, R. Poranne, and S. Coros, "Dynamic manipulation of deformable objects with implicit integration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4209–4216, 2021.

[10] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, pp. 239–249, 2020.

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, eabc5986, 2020.

[12] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Iterative residual policy: For goal-conditioned dynamic manipulation of deformable objects," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 389–404, 2024.

[13] S. Adebola, T. Sadjadpour, K. El-Refai, W. Panitch, Z. Ma, R. Lin, T. Qiu, S. Ganti, C. Le, J. Drake, and K. Goldberg, "Automating deformable gasket assembly," in *2024 IEEE 20th International Conference on Automation Science and Engineering*, 2024, pp. 4146–4153.

[14] R. Lee, M. Hamaya, T. Murooka, Y. Ijiri, and P. Corke, "Sample-efficient learning of deformable linear object manipulation in the real world through self-supervision," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 573–580, 2022.

[15] Y. Lian, X. Zhang, and Y. Liu, "Coupling of finite element method with material point method by local multi-mesh contact method," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 47-48, pp. 3482–3494, 2011.

[16] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1998, p. 43–54.

[17] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw, "Robust quasistatic finite elements and flesh simulation," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2005, pp. 181–190.

[18] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder, "Discrete shells," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003, pp. 62–67.

[19] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, eabd8803, 2021.

[20] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," *ACM Transactions on Graphics*, vol. 27, no. 3, p. 1–12, 2008.

[21] S. Duenser, R. Poranne, B. Thomaszewski, and S. Coros, "Robocut: Hot-wire cutting with robot-controlled flexible rods," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 98–1, 2020.

[22] M. Teschner, B. Heidelberger, M. Muller, and M. Gross, "A versatile and robust model for geometrically complex deformable solids," in *Proceedings Computer Graphics International, 2004.* IEEE, 2004, pp. 312–319.

[23] G. Irving, J. Teran, and R. Fedkiw, "Invertible finite elements for robust simulation of large deformation," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004, pp. 131–140.

[24] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[25] A. C. Pipkin, "The relaxed energy density for isotropic elastic membranes," *IMA journal of applied mathematics*, vol. 36, no. 1, pp. 85–99, 1986.

[26] D. Steigmann, "Tension-field theory," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 429, pp. 141–173, 1990.

[27] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: Analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics*, vol. 39, no. 6, 2020.

[28] S. Zimmermann, R. Poranne, and S. Coros, "Optimal control via second order sensitivity analysis," 2019, arXiv:1905.08534.

[29] M. Li, Z. Ferguson, T. Schneider, T. R. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: intersection-and inversion-free, large-deformation dynamics." *ACM Transactions on Graphics*, vol. 39, no. 4, p. 49, 2020.

[30] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, 2012, pp. 137–144.

[31] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on Robotics*, 2023.

[32] J. Envall, R. Poranne, and S. Coros, "Differentiable task assignment and motion planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2023, pp. 2049–2056.

[33] S. Gowda, Y. Ma, A. Cheli, M. Gwóźdź, V. B. Shah, A. Edelman, and C. Rackauckas, "High-performance symbolic-numerics via multiple dispatch," *ACM Communications in Computer Algebra*, vol. 55, no. 3, pp. 92–96, 2022.

[34] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in julia," 2016, arXiv:1607.07892.

[35] S. Zimmermann, M. Busenhart, S. Huber, R. Poranne, and S. Coros, "Differentiable collision avoidance using collision primitives," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2022, pp. 8086–8093.

[36] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning." in *International Joint Conference on Artificial Intelligence*, 2015, pp. 1930–1936.

[37] V. N. Hartmann, O. S. Oguz, D. Driess, M. Toussaint, and A. Menges, "Robust task and motion planning for long-horizon architectural construction planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2020, pp. 6886–6893.

[38] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.