Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting

Eric Heiden*†, Miles Macklin*, Yashraj Narang*, Dieter Fox*§, Animesh Garg*‡, Fabio Ramos*¶
*NVIDIA, USA
†University of Southern California, Los Angeles, USA. Email: heiden@usc.edu
‡University of Toronto & Vector Institute, Canada
§University of Washington, Seattle, USA
¶University of Sydney, Australia

***Abstract*—Robotic cutting of soft materials is critical for applications such as food processing, household automation, and surgical manipulation. As in other areas of robotics, simulators can facilitate controller verification, policy learning, and dataset generation. Moreover, *differentiable* simulators can enable gradient-based optimization, which is invaluable for calibrating simulation parameters and optimizing controllers. In this work, we present DiSECt: the first differentiable simulator for cutting soft materials. The simulator augments the finite element method (FEM) with a continuous contact model based on signed distance fields (SDF), as well as a continuous damage model that inserts springs on opposite sides of the cutting plane and allows them to weaken until zero stiffness, enabling crack formation. Through various experiments, we evaluate the performance of the simulator. We first show that the simulator can be calibrated to match resultant forces and deformation fields from a state-of-the-art commercial solver and real-world cutting datasets, with generality across cutting velocities and object instances. We then show that Bayesian inference can be performed efficiently by leveraging the differentiability of the simulator, estimating posteriors over hundreds of parameters in a fraction of the time of derivative-free methods. Finally, we illustrate that control parameters in the simulation can be optimized to minimize cutting forces via lateral slicing motions. We publish videos and additional results on our project website at https://diff-cutting-sim.github.io.**

I. Introduction

Robotic cutting of soft materials is critical for various real-world applications, including food processing, household automation, surgical manipulation, and manufacturing of deformable objects. As in other areas of robotics, simulators can allow researchers to verify controllers, train control policies, and generate synthetic datasets for cutting, as well as avoid expensive or time-consuming real-world trials. In addition, cutting is inherently destructive and irreversible; thus, accurate and efficient simulators are indispensable for automating safety-critical tasks like robotic surgery.

However, simulating the cutting of soft materials is challenging. Cutting involves diverse physical phenomena, including contact, friction, elastic deformation, damage, plastic deformation, crack initiation, and/or fracture. A variety of simulation methods have been introduced, including mesh-based approaches (e.g., extensions of the finite element method (FEM)), particle-based approaches (e.g., smoothed particle hydrodynamics (SPH)), and hybrid techniques. For accuracy, these methods often require computationally-expensive remeshing, or simulation of millions of particle interactions.

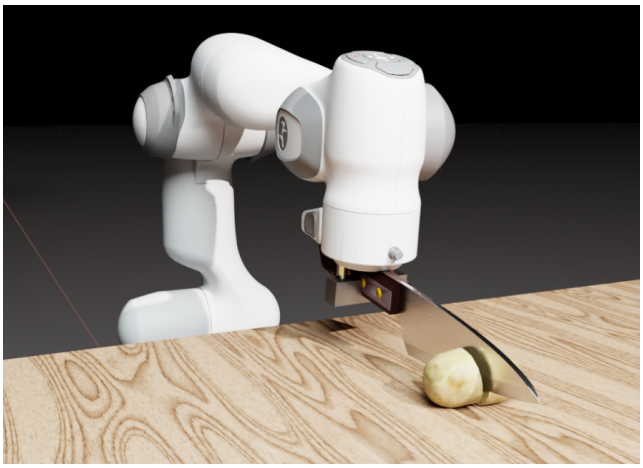

Fig. 1: A rendering from our differentiable cutting simulator. DiSECt provides accurate gradients of the cutting process, allowing us to efficiently fit model parameters to real-world measurements, and optimize cutting motions.

Moreover, a physically-accurate forward simulator is necessary but not sufficient for accurate and useful predictions. The material properties of real-world soft materials (e.g., ripening fruits, diseased tissue) are often unknown and highly heterogeneous, mandating calibration of material models. In addition, ideal trajectories for cutting may not be known beforehand, requiring efficient optimization of control actions. In other fields, the need to infer material and control parameters has motivated the development of differentiable simulators, which can harness efficient, gradient-based optimization methods.

We present DiSECt, the first differentiable simulator for the cutting of soft materials (Figure 1). The simulator has several key features. First, contact between the cutting instrument (i.e., knife) and the object is resolved using a continuous signed distance field (SDF) representation of the knife. Second, a mesh-based representation of the object is used; given a predefined cutting plane, virtual nodes are inserted along the surface in a preprocessing step that occurs only once. Third, cracks are introduced using a continuous damage model, where springs are inserted on either side of the cutting plane. Under

application of force over time, the springs can progressively weaken until zero stiffness, producing a crack. The continuous contact and damage models enable differentiability, and the springs and particles provide hundreds of degrees of freedom that can be used to calibrate the simulator against ground-truth data. Gradients of any simulation parameter are computed using automatic differentiation via source-code transformation.

This work offers the following contributions:

1) The first differentiable simulator for cutting soft materials.
2) A comparison of gradient-based and derivative-free methods for calibrating the simulator against ground-truth data from commercial solvers and real-world datasets. The comparison demonstrates that the differentiability of the simulator enables highly efficient estimation of posteriors over hundreds of simulation parameters.
3) A performance evaluation of the calibrated simulator for unseen cutting velocities and object instances. It demonstrates that the simulator can accurately predict resultant forces and nodal displacement fields, with simulation speeds that are orders of magnitude faster than a comparable commercial solver.
4) An application of the simulator to robotic control, optimizing knife motion trajectories to minimize cutting forces under time constraints.

II. Related Work

A. Modeling and Simulation of Cutting

Analytical modeling: Cutting is a branch of elastoplastic fracture mechanics [2], where theoretical analysis has primarily focused on metal cutting [56] and brittle materials [23]. A comprehensive treatment of the mechanics involved in cutting of metals, biomaterials and non-metals is given in Atkins and Atkins [2]. Analytical models of the forces acting on a knife as it cuts through soft materials have been derived in [15, 63].

Mesh-based simulation: Among the numerical methods that implement such analytical models, the Finite Element Method (FEM) [5, 32] is a commonly used technique to simulate deformable bodies. FEM solves partial differential equations over a given domain (defined by a mesh) by discretizing it into simpler elements, such as tetrahedra (which we use in this work). Without modifying the mesh topology, the Extended FEM (X-FEM) [58] augments mesh elements by enrichment functions to model fracture mechanics processes [35, 36, 40].

In classical FEM, topological changes that result from cutting and other fracture processes mandate an adaption of the mesh resolution so that the propagation of the crack can be accurately simulated. Approaches in mechanical engineering [1] and computer graphics [7, 9, 39, 69, 90] have been introduced that re-mesh the simulation domain to accommodate cuts and other forms of cracks.

Virtual node algorithms (VNA) [59, 75, 83] duplicate mesh elements that intersect with the cutting surface, resulting in elements with portions of real material and empty regions. At the cutting interface, virtual nodes are introduced that allow for a two-way coupling of contact and elastic forces with the underlying mesh of the separated parts. We leverage VNA to augment our FEM simulation with extra degrees of freedom to allow the fine-grained simulation of contact between the material and the knife. Furthermore, by augmenting the mesh only once at the beginning of the cut, we avoid discontinuous re-meshing operations and are able to compute gradients from our simulator.

Mesh-free and hybrid approaches: So-called mesh-free Lagrangian methods, such as smoothed-particle hydrodynamics (SPH) [21, 60], element-free Galerkin (EFG) [4], and smoothed-particle Galerkin (SPG) [89], simulate continuous media by many particles that cover the simulation domain and the dynamics is determined by a kernel that defines the interaction between spatially close particles. Hybrid approaches have been proposed, such as the Material Point Method (MPM) [29, 82, 86, 87], that combine Eulerian (grid-based) and Lagrangian (particle-based) techniques. Position-based Dynamics (PBD) [64] has been explored to simulate cutting in interactive surgery simulators [6, 67].

Robotic cutting: In [47] a robotic meat-cutting system is introduced that scans objects online to simulate deformable objects and uses impedance control to steer the knife. In [41, 78], a simplified mass-spring model is used for learning to cut planar surfaces. Such trained policies are then transferred to a physical robotic surgery system. By studying the mechanics of cutting, [63] derive control strategies involving pressing, pushing, and slicing motions. Similarly, [15, 16] analyze cutting "by pressing and slicing," while taking into account the blade sharpness of the knife. Constrained optimization is used in [85] to optimize cutting trajectories while accounting for contact forces, whereas in our control experiment we furthermore account for the coupling of elastic and contact force between the knife and the deformable object being cut. In Jamdagni and Jia [34], FEM simulation for robotic cutting is devised where crack propagation is simulated at the cross sections on a high-resolution 2D mesh, which is coupled with a coarser 3D mesh simulation. Meshes are obtained from laser scans of biomaterials, and force profiles recorded from a force sensor attached to a robot end-effector as it vertically cuts through various foodstuffs. In this work, we leverage this dataset of real-world cutting trajectories. Fully data-driven models have been learned to facilitate model-predictive control in robotic cutting [42, 57].

B. Differentiable Simulation

Differentiable simulation has gained increasing attention, as it allows for the use of efficient gradient-based optimization algorithms to tune simulation parameters or control policies [10, 14, 19, 20, 25, 26, 31, 38, 65, 72]. Although finite differencing may be used to approximate the derivative of a simulation output, this approach suffers from accuracy problems and does not scale to large numbers of parameters [51]. In this work we employ reverse-mode automatic differentiation

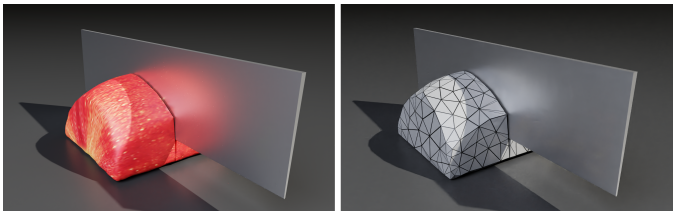

Fig. 2: Visualization of an apple slice. We use a tetrahedral FEM-based model of the apple generated from scanned real-world data [34].

to obtain gradient information via source-code transformation [22, 33]. Recent work has shown the potential for source-code transformation to generate efficient parallel kernels for reverse-mode differentiation using graphics-processing units (GPUs) [30, 31].

C. Parameter Inference for Simulators

Simulation-based inference is a methodology that has emerged across various fields of science [12]. In parameter identification, optimization-based approaches are often utilized to obtain point estimates of the simulation parameters (e.g., for constitutive equations [24, 50]) that minimize the model error as measured from the dynamics of the real system. More often, when analytical gradients are unavailable or too expensive to obtain, optimization with finite differencing or gradient-free methods have been applied [55, 66], particularly for dynamical systems through system identification [46].

Probabilistic inference techniques, on the other hand, seek to infer a distribution of simulation parameters that allows downstream applications to evaluate the uncertainty of the estimates. Such methods have been applied to learn conditional densities of simulation parameters given trajectories from the simulator and the real system [27, 52, 53, 68, 73, 80].

Parameter inference may also be integrated in a closed-loop system [11, 54, 62, 73], where the currently estimated posterior over simulation parameters guides domain randomization and policy learning. Such approaches iteratively reduce the sim2real gap.

III. DiSECt: Differentiable Simulator for Cutting

The design of our simulator for cutting biomaterials is motivated by three aspects:

1) The model has to be physically plausible such that the effects of changing physical quantities from continuum and fracture mechanics can be observed realistically.
2) While the cutting process is an inherently discontinuous operation, wherever possible, gradients of the simulation parameters must be efficiently obtainable.
3) The simulator must allow for sufficient degrees of freedom so that fine differences in material properties can be identified at localized places where the knife cuts through a heterogeneous material.

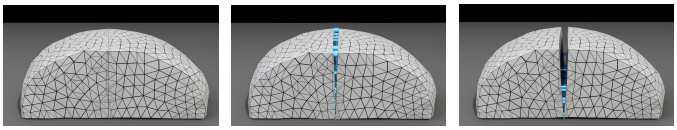

Fig. 3: To smoothly model damage, we insert springs (shown in blue) between cut elements. We incrementally reduce the spring stiffness based on damage over time, proportional to the contact force applied from the knife. From left at $t = 0\,\text{s}$, $t = 0.8\,\text{s}$, $t = 1.2\,\text{s}$. Here, we have removed the knife from visualization to show the inserted springs clearly.

A. Continuum Mechanics

We implement the Finite Element Method (FEM) to simulate the dynamics of the soft materials used throughout this work. Based on a tetrahedral discretization of the cutting target object, elastic forces are computed that take into account material properties, such as Young's modulus, Poisson's ratio, and density.

We employ a Neo-Hookean constitutive model following the strain-energy density function from Smith et al. [77], which has been designed to model biological tissue and preserve volume when the mesh undergoes large deformations:

$$\Psi = \frac{\mu}{2}(I_C - 3) + \frac{\lambda}{2}(J - \alpha)^2 - \frac{\mu}{2}\log(I_C + 1). \qquad (1)$$

Here λ, μ are the Lamé parameters[1] and α is a constant. $J = \det(\mathbf{F})$ is the relative volume change, $I_C = \text{tr}(\mathbf{F}^T\mathbf{F})$ is the first invariant of the Cauchy-Green deformation tensor, and $\mathbf{F}$ is the deformation gradient. We give the material properties for our experimental objects in Table A1. Integrating over each tetrahedral element and summing the energy from (1) over all elements gives the total elastic potential energy. Forces $\mathbf{f}_{\text{elastic}}$ are derived from the energy gradient analytically and integrated using a semi-implicit Euler scheme. In addition to elastic forces, we include a strain-rate dissipation potential to model internal damping [48].

B. Mesh Pre-Processing

Before the simulation begins, the cutting surface for the entire cut is given as a triangle mesh. We implement the Virtual Node Algorithm from Sifakis et al. [75], which, as shown in Figure 4, duplicates the mesh elements (tets) that intersect with the cut (subfigure 2), so that the tet above the cutting surface has a portion of material above the cut, and one empty portion below the cut (and vice versa for the reciprocal element). Next, we insert virtual nodes at the intersection points on the edges. These nodes are only represented by their barycentric coordinate $u \in [0, 1]$.

Similar to the two-way coupled "soft particles" described in Sifakis et al. [76], each virtual node's 3D position $\tilde{\mathbf{x}}$ and velocity $\tilde{\mathbf{v}}$ is entirely defined by the coordinates of its two parent vertices, indexed i and j:

$$\tilde{\mathbf{x}} = (1-u)\mathbf{x}_i + u\mathbf{x}_j \qquad \tilde{\mathbf{v}} = (1-u)\mathbf{v}_i + u\mathbf{v}_j.$$

[1]Lamé parameters are a reformulation of Young's modulus and Poisson's ratio; hence we use these terms interchangeably.

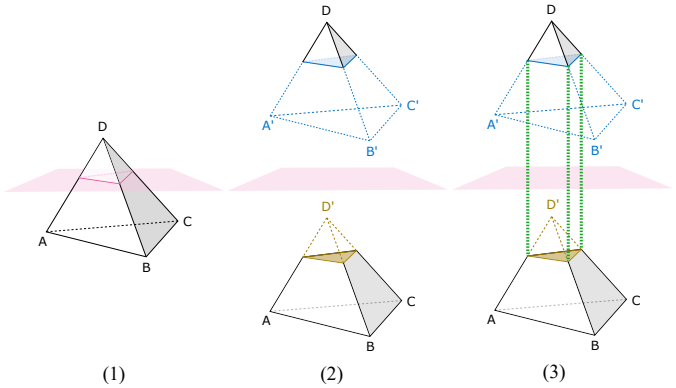

Fig. 4: Pre-processing of a tetrahedral mesh (visualized by a single tet) prior to being cut along a given cutting surface (red plane). Given the original mesh, (1) intersecting elements are identified, and (2) duplicated such that the intersection geometry is retained in each part of the cut elements. Virtual nodes are inserted where the edges are intersecting the cutting surface. (3) The original and duplicated elements are connected by springs (green) between the virtual nodes.

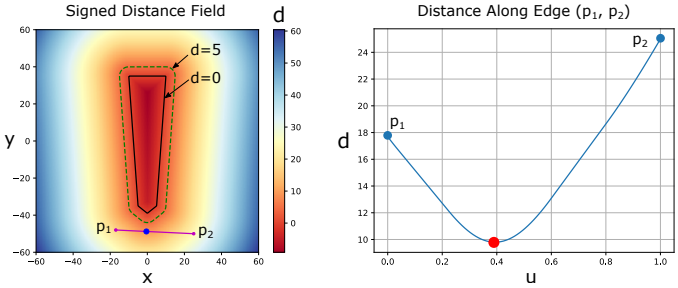

Fig. 5: 2D slice of the signed distance field (SDF) for the knife shape (not true to scale), where the color indicates distance d. The knife's boundary at $d = 0$ is indicated by a solid black contour line. *Left:* at distances greater than zero the SDF becomes more rounded. *Right:* distance d along the edge (p_1, p_2) with barycentric edge coordinate u varying between 0 (p_1) and 1 (p_2). The closest point found by Algorithm 2 is shown in red. Distances are exemplary and do not represent the actual dimensions used in the simulator (see Appendix A2).

The edge sections along the non-empty portions of the duplicated mesh elements participate in contact dynamics and propagate the resulting contact forces back to their parent nodes, as described in III-C. Edge sections from the empty portions of the mesh create no collisions and are solely updated from the FEM dynamics of the parent vertices.

In the final step of this mesh preprocessing phase (subfigure 3), springs are inserted that connect the virtual nodes on both sides of the cutting surface which originally belonged to the same edge of the mesh. These springs allow us to simulate damage occurring during the cutting process in an entirely continuous (and thereby differentiable) manner, by weakening their stiffness values as knife contact forces are applied over time.

While our approach in its current form assumes that the entire cutting surface is given before the cutting simulation begins, interactive cutting applications could be accommodated by interweaving this mesh augmentation step with the actual cutting simulation. Nonetheless, the practicality of using such an interactive approach for parameter inference remains to be validated.

C. Contact Dynamics

Following Macklin et al. [49], we implement a contact model that represents the knife shape by a signed distance function (SDF) (Figure 5) that interacts with the tetrahedral mesh of the object being cut. To find the closest point between an edge from the mesh and the SDF, we run 20 iterations of the Frank-Wolfe algorithm (Algorithm 2 in appendix), that uses the gradient information from the SDF to find a locally optimal solution for the barycentric coordinate $u \in [0,1]$ with the smallest distance. Using this coordinate, we can compute the penetration depth and contact normal by querying the SDF and its gradient. Penalizing collisions, the contact normal force is computed as the squared penetration depth in the direction of the normal (zero if no collision). In combination with the relative velocity between the knife and the mesh vertices, friction forces are computed following the continuous friction model from Brown [8, Equation 4.5].

Analogous to the knife-mesh contact dynamics, we simulate contact forces between the object mesh and the ground that it rests on via the same penalty-based contact model. Here, the forces are computed between mesh vertices, represented by spheres as collision geometry, and the ground represented by a half space. To prevent the object from sliding off the table during the cut, as in [34], we apply boundary conditions that fix mesh vertices in place when they fulfill both the following conditions: (1) touching the ground and (2) being located 1 cm away from the cutting plane.

D. Damage Mechanics

Physically, damage refers to a macroscopic reduction in stiffness or strength of a material caused by the formation and growth of microscopic defects (e.g., voids and microcracks). For fruits and vegetables, which often have limited plastic deformation regimes before failure, damage can be approximated by a reduction in the elastic modulus of the material, or in a discrete mesh-based formulation, in the components of the stiffness matrix.

As follows, our model for damage mechanics leverages the springs that have been introduced in the final step in subsection III-B. As the knife applies force to the cutting interface, the stiffnesses of the springs that are in contact with the knife, and hence receive knife contact forces, are linearly decreased (see a visualization of this progressive weakening as the knife slides down the cutting interface in Figure 3):

$$k'_e = k_e - \gamma \|\mathbf{f}_{\text{knife}}\|, \tag{2}$$

where $\mathbf{f}_{\text{knife}}$ is the force the knife applies on the spring (i.e., the contact force that is computed between the knife and the edge), and $\gamma \in [0,1]$ is a coefficient that controls the "weakness" of the spring (i.e., how easily the material weakens and separates as the knife applies force to it).

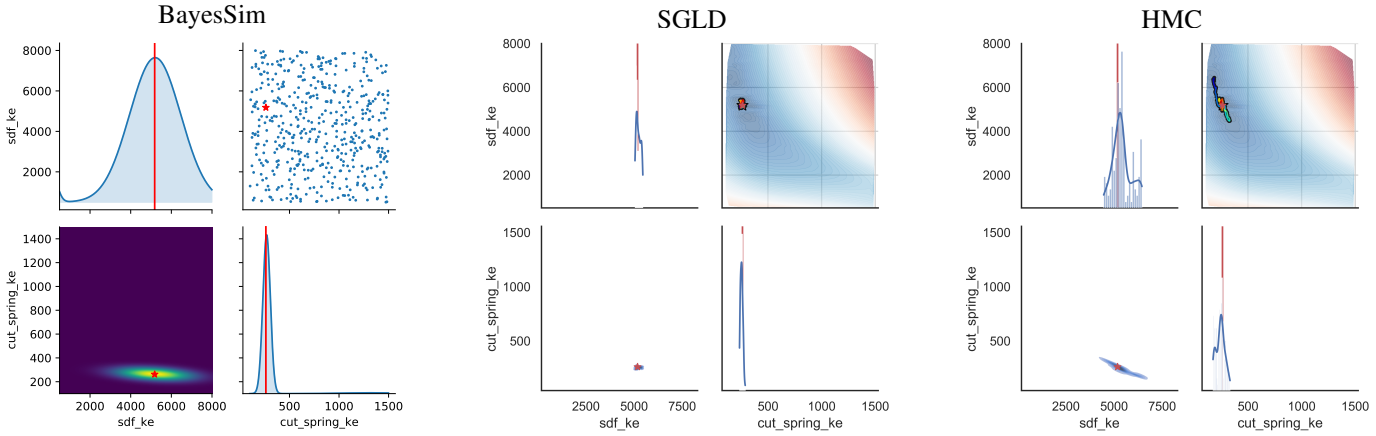

Fig. 6: Parameter posterior inferred by BayesSim (left), Stochastic Gradient Langevin Dynamics (SGLD) after 90 burn-in iterations (center), and Hamiltonian Monte Carlo (HMC) after 50 burn-in iterations (right) for the two-dimensional parameter inference experiment from subsection V-A, in which cutting spring stiffness `sdf_ke` and knife contact force stiffness `sdf_ke` need to be estimated. The diagonals of each figure show the marginal densities for the two estimated parameters. The bottom-left sections show a heatmap of the joint distribution. The top-right scatter plot of the BayesSim figure shows the sampled parameters from the training dataset (blue) and the ground-truth (red). The top-right plot sections for SGLD and HMC visualize the loss landscape as a heatmap (where blue means smaller error than red), with the Markov Chain depicted by a colored line. Red lines and stars indicate ground-truth parameter values. SGLD and HMC leverage the gradients of our differentiable simulator and show a much sharper posterior distribution than BayesSim, which uses a dataset of 500 simulated force profiles.

Algorithm 1: Simulation Loop in DiSECt

for $i = 1 \ldots$ timesteps **do**

1. Compute gravity and external contact forces $\mathbf{f}_{\text{ext}}$ between mesh and cutting board (half space).
2. Compute elastic forces $\mathbf{f}_{\text{elastic}}$ following the constitutive model in subsection III-A.
3. Compute knife contact forces $\mathbf{f}_{\text{knife}}$ as described in subsection III-C.
4. Update cutting spring stiffness $k_e' = k_e - \gamma \|\mathbf{f}_{\text{knife}}\|$.
5. Compute cutting spring forces $\mathbf{f}_{\text{spring}}$.
6. Semi-implicit Euler integration of mesh vertices:
   $\mathbf{v}^{t+1} \leftarrow \mathbf{v}^t + \Delta t \mathbf{M}^{-1}(\mathbf{f}_{\text{knife}} + \mathbf{f}_{\text{spring}} + \mathbf{f}_{\text{elastic}} + \mathbf{f}_{\text{ext}})$
   $\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \Delta t \mathbf{v}^{t+1}$
7. Euler integration of knife velocity (prescribed velocity trajectory).

end

We use a simulation time step of $\Delta t = 10^{-5}$s across all our experiments to accommodate the simulation of and stiff materials (such as apples or potatoes) and centimeter-scale meshes obtained by laser scans of real biomaterials. Our full simulation loop is described in Algorithm 1.

IV. Simulation Parameter Inference

DiSECt supports two modes of parameterizing the simulator: the cutting spring parameters shown in Appendix Table A2 can be shared across all springs, or tuned individually per spring. The shared parameterization is low-dimensional since only one scalar per parameter type needs to be inferred. In contrast, when the cutting springs are individually parameterized, hundreds of variables need to be tuned.

While it is often possible to hand-tune parameters of low-dimensional, phenomenological models, the task of identifying the simulation parameters that can enable close prediction of real-world measurements is daunting, particularly for complex models such as ours with individually tuned spring parameters. To tackle the simulation calibration problem, we leverage automatic differentiation and GPU acceleration to efficiently compute gradients for all the parameters of our simulator. This allows us to use optimization techniques, such as stochastic gradient descent, to directly compute point estimates for the parameters. Moreover, we can leverage modern Bayesian inference methods and estimate posterior distributions for the high-dimensional parameter set given physical observations, such as the force profile of the knife while cutting real foodstuffs. The result is a simulator with the capacity to identify its own uncertainty about the physical world, leading to more robust simulations. We follow a conventional Bayesian approach and define $p(\theta)$ as the prior distribution over simulation parameters (see Appendix Table A2) which, in our experiments, is a uniform distribution, and $p(\phi^r|\theta)$ as the likelihood function given by $p(\phi^r|\theta) = \exp\{-\|\phi_\theta^s - \phi^r\|_L\}$, where ϕ^r corresponds to real trajectory observations such as knife forces, positions and velocities. ϕ_θ^s is the equivalent simulated trajectory, and $\|\cdot\|_L$ is the L-norm. In our experiments we use the $L1$, which we determined the most effective (see Appendix A-B). With both prior and likelihood functions, we can compute the posterior as $p(\theta|\phi^r) = \frac{1}{Z} p(\phi^r|\theta)p(\theta)$, where Z is a normalizing constant, also known as the marginal likelihood.

In the next sections, we describe three techniques for simulator parameter inference. We start with a gradient-based approach that produces point estimates, followed by stochastic gradient Langevin dynamics (SGLD), a popular gradient-

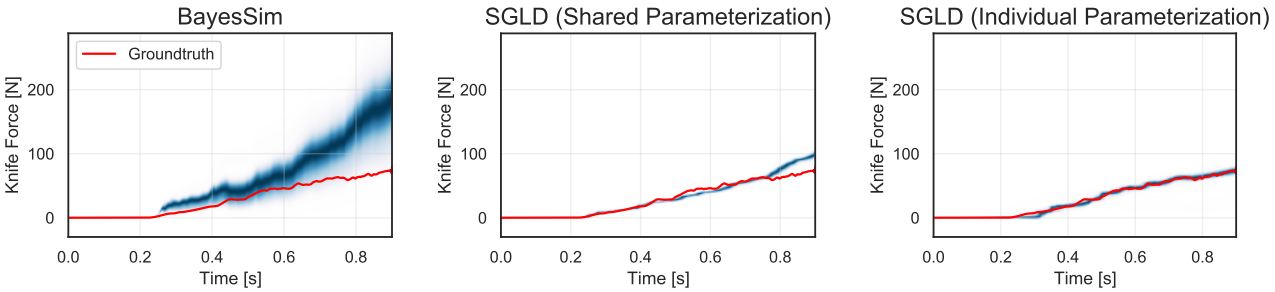

Fig. 7: Kernel density estimation from 20 knife force trajectories rolled out by sampling from the posterior found by BayesSim (left) and SGLD applied in our differentiable simulator with shared (center) and individual parameterization (right). Shown in red is the ground-truth trajectory from a commercial simulation of cutting an apple with a hemispherical shape. Areas of higher density are shaded in dark blue.

based Markov chain Monte Carlo technique that approximates the posterior as a set of particles. Finally, we describe the likelihood-free inference technique known as BayesSim, which does not make use of gradients and is used as a baseline, demonstrating the value of differentiable simulation.

A. Gradient-Based Optimization

As a baseline for probabilistic parameter inference, we present a solution based on stochastic gradient descent using the popular Adaptive Moment Estimation (Adam) optimizer [37] (see Appendix A-A), a first-order method that scales the parameter gradients with respect to their running averages and variances. Unlike the Monte Carlo posterior approximation obtained by SGLD, Adam will find a locally optimal point estimate to the parameters that minimizes the expected loss. We define the loss as $l(\theta) = \log p(\phi^r|\theta)$ and compute gradients with respect to the simulation parameters θ that minimize the loss between a real trajectory ϕ^r and simulated trajectories ϕ^s.

B. Stochastic Gradient Langevin Dynamics

A popular alternative to Adam for probabilistic inference is the stochastic gradient Langevin dynamics (SGLD) method [84] (Algorithm 4). SGLD combines the benefits of having access to parameter gradients with well-established sampling-based methods for probabilistic inference to significantly scale the parameter set to high dimensions at a tractable computational cost. The method can be seen as an iterative stochastic gradient optimization approach with the addition of Gaussian noise which is scaled by a preconditioner factor at every iteration. Given a sequence of trajectories generated by our simulator and a sequence of observed trajectories, $\Phi = \{\phi_i^s, \phi_i^r\}_{i=1}^N$, we can write the posterior distribution as

$$p(\theta|\Phi) \propto p(\theta)\prod_{i=1}^{N} p(\phi_i^r|\theta). \tag{3}$$

SGLD takes the energy function of the posterior denoted by $U(\theta) = -\sum_{i=1}^{N}\frac{1}{N}\log p(\Phi|\theta) - \log p(\theta)$ and samples from the posterior using the following rule:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{2}A(\theta_t)\nabla U(\theta_t) + \eta_t, \tag{4}$$

where $\eta_t \sim \mathcal{N}(0, A(\theta_t)\alpha)$, α is the learning rate, and A is a preconditioner. After an initial burn-in phase necessary for the Markov chain to converge, m samples can be stored to recover an approximate posterior given by $p(\theta|\Phi) \approx \frac{1}{m}\sum_{i=1}^{m}\delta_{\theta_i}(\theta)$, where $\delta_{\theta_i}(x)$ is the Dirac delta function which is non-zero whenever $x = \theta_i$.

Critical to SGLD's performance is the choice of an appropriate preconditioner. In this work we follow the extension proposed in [43] that computes the preconditioner as an approximation of the Fisher information matrix of the posterior distribution given by A. This approximation can then be sequentially updated using the gradient of the energy function. This is a similar process as the popular RMSProp [79]. Specifically, the preconditioner $A \in \mathbb{R}^{m\times m}$ and momentum $V \in \mathbb{R}^m$ are updated as

$$V(\theta_t) = \beta V(\theta_{t-1}) + (1-\beta)\nabla U(\theta_t) \odot \nabla U(\theta_t), \tag{5}$$

$$A(\theta_t) = \mathrm{diag}\left(1 \oslash \left(\varepsilon + \sqrt{V(\theta_t)}\right)\right), \tag{6}$$

where $\varepsilon > 0$ is a small diagonal bias (we choose $\varepsilon = 10^{-8}$) added to the preconditioner to prevent it from degenerating, and β (which we set to 0.95) is the exponential decay rate of the preconditioner. $\odot$ and $\oslash$ are element-wise multiplication and division operators.

C. Likelihood-free Inference via BayesSim

BayesSim is a likelihood-free technique to estimate the parameters of a derivative-free simulator that implements a forward dynamics model which is used to generate trajectories for the given simulation parameters. The technique consists of learning a conditional density $q(\theta|\phi)$ (which BayesSim represents as a Gaussian mixture model), where θ are the simulation parameters, and ϕ are trajectories or summary statistics of trajectories. For details about our particular BayesSim implementation, please see Appendix A-C.

V. Experiments

To evaluate DiSECt, we first use synthetic data from our own simulator to evaluate probabilistic inference algorithms. Next, we identify the simulation parameters (Appendix Table A2) to closely match an industry-standard, high-fidelity

simulation where we have access to the nodal forces and precise motion of the vertices resulting from the knife contact. Finally, we leverage real-world experimental data of measured knife force profiles to evaluate our sim2real transfer.

A. Parameter Inference from Synthetic Data

In our first experiment, we investigate how accurate the estimated posterior is, given synthetic force profiles from known simulation parameters. We create a dataset of 500 knife force trajectories by varying two of the parameters using the shared parameterization (section IV) from our proposed simulator as training domain. We choose such low dimensionality to ensure that we can obtain enough training data to sample from the uniform prior distribution which is crucial to the performance of likelihood-free methods, such as BayesSim.

The two parameters to be estimated are `sdf_ke` (ranging from 500 to 8000), the stiffness of the contact model at the mesh surface, and `cut_spring_ke` (ranging from 100 to 1500), the stiffness of the cutting springs (cf. subsection III-D) at the beginning of the simulation.

As shown in Figure 6, BayesSim captures the posterior (see heatmap in the lower-left corner of the left subfigure) around the ground-truth parameters of `sdf_ke` = 5100 and `cut_spring_ke` = 200. SGLD (center), however, yields a significantly sharper density estimate around the true values within 90 burn-in iterations, while Hamiltonian Monte-Carlo (HMC), another gradient-based Bayesian estimation algorithm, finds a slightly wider posterior within approximately 50 iterations.

B. Parameter Inference from High-fidelity Simulator

In our first set of experiments where the training data does not stem from our own model, we simulate cutting trajectories using a commercial, explicit dynamics simulator as a ground-truth source. Details on the simulation setup are in Appendix B-B. For each simulation, knife force and nodal motion trajectories were extracted. Each simulation was executed across 4 CPUs and took an average of 1941 min (>32 h) to complete. For comparison, our simulator produces a cutting trajectory of 1 s duration (with a 1×10^{-5} s simulation time step) within 30 s on an NVIDIA RTX 2080 GPU, and the gradient of the cost function (likelihood) within 90 s.

We estimate the following parameters θ in this experiment: (1) `sdf_ke`, (2) `sdf_kd`, (3) `sdf_kf`, (4) `sdf_mu`, (5) `cut_spring_ke`, (6) `cut_spring_softness`, and (7) `initial_y` (for an explanation see Table A2 in the appendix).

First, we train BayesSim in an iterative procedure where the currently estimated posterior over the simulation parameters is sampled to roll out a new set of trajectories that are added to the training dataset. Starting from 500 trajectories, we repeatedly sample 20 new parameters and refit BayesSim's mixture density network (MDN) to the updated training dataset. After 100 such iterations (i.e., 2000 additional roll-outs), we obtain the posterior shown in appendix Figure A7.

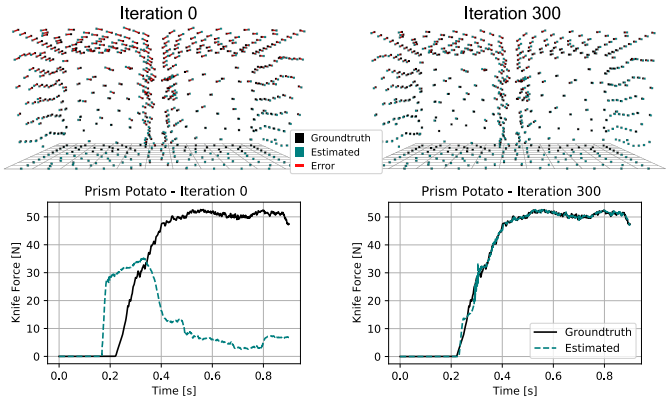

Fig. 8: Results from simulation parameter optimization given the positions of the vertices (top row) and the knife force (bottom row) with a commercial simulation as ground-truth. *Left:* before optimization, the vertices (top) at the last time step (0.9 s) of the trajectory are visibly distinct between our simulation (blue) and the ground-truth (black), as shown by the red lines indicating the vertex difference. *Right:* after 300 steps with the Adam optimizer, the vertices (top), as well as the knife force profile (bottom), are closer after the parameter inference.

For comparison, we train SGLD in DiSECt for 300 iterations with the same parameterization as used for BayesSim, i.e., the parameters are represented by scalars that are shared across all cutting springs. As shown in Figure 7, trajectories sampled from the estimated posterior of SGLD result in a significantly closer fit (center) to the ground-truth knife force profile compared to BayesSim (left). The average mean absolute error (MAE) of the roll-outs with 4.714 N significantly outperforms BayesSim's average MAE of 26.860 N. The gradient-based estimation method achieves such an outcome with less training data, as it only took 300 trajectory simulations, compared to the 2500 trajectories in total that served as the training dataset for BayesSim. If we allow the parameters to be optimized individually for each cutting spring (resulting in 1737 parameters in total), the resulting simulation becomes even closer (right), with an average MAE of 3.075 N.

Based on the commercial ground-truth simulation, we collect additional object nodal displacement field trajectories in addition to the knife force profiles, which allows us to leverage another reference signal to calibrate DiSECt. At 18 reference time steps within the trajectory roll-out, we compute the $L2$ error between the positions of the vertices in the ground-truth and our simulator, and add it to the overall cost (which previously only consisted of the $L1$ norm over knife force difference) with a tuned weighting factor. By optimizing the aforementioned parameters individually with the new cost function through Adam, after 300 iterations, we arrive at a simulation that not only closely matches the knife force profile from the ground-truth simulation, but also has a significantly reduced gap in the nodal motions between the two simulators (see Figure 8). Before the optimization, the mean Euclidean distance between the simulated nodes and the ground-truth nodal positions is 1.554 mm at the last time step ($t = 0.9$ s).

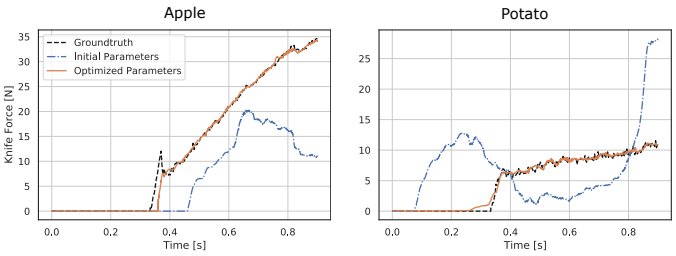

Fig. 9: Results from optimizing simulation parameters in DiSECt given real-world knife force profiles from vertical cutting of an apple (left) and a potato (right).

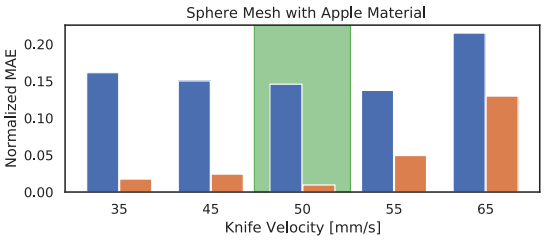

Fig. 10: Velocity generalization results for a sphere shape with apple material properties given ground-truth simulations with different vertical knife velocities from a commercial simulator. Two versions of DiSECt were calibrated: by sharing the parameters across all cutting springs (blue) and by optimizing each value individually (orange), given a ground-truth trajectory with the knife sliding down at $50\,\mathrm{mm\,s^{-1}}$ speed (highlighted in green). The normalized mean absolute error (MAE) is evaluated against the ground-truth by rolling out the estimated parameters for the given knife velocity.

The low initial error over the nodal motion is explained by the fact that the material properties have been set to already match a potato (Table A1), leaving the cutting-related spring parameters as the only remaining variables to be estimated. After 300 iterations, this error reduced to 1.289 mm, while the knife force profile MAE decreased from 28.366 N to 0.549 N.

C. Parameter Inference from Real World Measurements

In this experiment, we calibrate DiSECt to real-world cutting trajectories. The real-world dataset provided by Jamdagni and Jia [34] contains 3D meshes created from laser scans of the actual objects being cut, as well as knife force profiles measured from a force sensor mounted between a robot end-effector and a knife. The knife dimensions are given in Table A2 and explained in Figure A3. The triangular surface meshes of the foodstuffs are discretized to tetrahedral meshes via the TetWild [28] meshing library. After optimizing the simulation parameters with Adam for 300 iterations, the simulator closely matches the force profile of a knife cutting an apple (Figure 9 left) with an MAE of 0.253 N, and for a potato cutting action (right) achieves an MAE of 0.379 N. We note that the optimization is stable without requiring restarts from different parameter configurations, and we found even relatively poor parameter initialization (such as the one shown in Figure 9) results in highly accurate prediction after the calibration.

D. Generalization

While our simulator is able to calibrate itself very closely to a variety of sources of ground-truth signals – whether these are knife force profiles obtained by a real robot cutting foodstuffs, or motion recordings from the mesh vertices in a commercial simulator that is entirely different from ours (see Appendix B-B) – the question of overfitting arises. In the following, we investigate how well the identified simulation parameters transfer to test regimes that differ from the training conditions under which these parameters were optimized.

1) Generalization to longer simulations: As shown in Table A2, DiSECt allows various dynamics parameters to be defined individually for each spring, resulting in hundreds of degrees of freedom. The larger parameterization provides more opportunity to find closely matching solutions, but may be prone to overfitting to the reference trajectory in certain settings. One of the pathological cases occurs when the goal is to predict the knife force trajectory for a duration longer than the time window seen during training from the reference force profile. In the experiment shown in Figure A4, we optimize the simulation parameters using Adam on the first 0.4 s section from the reference trajectory. Within that segment, the individual parameterization clearly outperforms the shared parameterization with a MAE of 0.306 N versus 0.384 N. However, when we test the estimated parameters on a simulation with a duration of 0.9 s, the shared parameterization achieves a closer fit with a 4.802 N MAE, compared to the individual tuning with a 5.921 N MAE. Intuitively, as the knife slices downward with constant velocity (at $50\,\mathrm{mm\,s^{-1}}$), fewer cutting springs are affected by the contact dynamics during a shorter roll-out since the knife does not progress far enough to reach the cutting springs closer to the ground. Hence, their parameters' gradients were zero during the estimation. Tuning the same kind of parameters uniformly allows all cutting springs to have an improved fit over the initialization, even when their interaction with the knife only becomes apparent at a later time.

2) Generalization to different knife velocities: We investigate how accurately DiSECt predicts knife force profiles given the parameters that were inferred from a cutting trajectory with a knife downward velocity of $50\,\mathrm{mm\,s^{-1}}$. At test time, we change the downward velocity to 35, 45, 55, and $65\,\mathrm{mm\,s^{-1}}$. As shown in Figure 10 (and Figure A5 in the appendix), the individual parameterization significantly outperforms the shared parameterization in normalized mean absolute error (NMAE), i.e., the MAE between the ground-truth and estimated trajectory divided by the mean force of the ground-truth trajectory, achieving a four-fold more accurate result compared to the shared parameterization in many cases (see example knife force profile in Figure 11).

3) Generalization to different geometries: Cutting the same type of biomaterial can lead to drastically different knife force profiles, even when all the variables that influence the motion of the knife remain the same [34]. This can be caused by different geometries even within the same object class, as

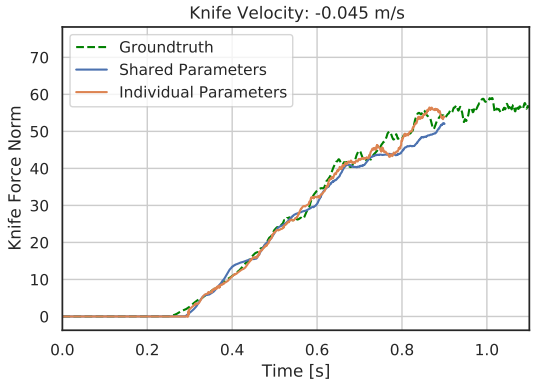

Fig. 11: Knife force profiles for cutting a cylindrical mesh with cucumber material properties by simulating the parameters with $45\text{mm}\,\text{s}^{-1}$ knife velocity downwards. The simulation parameters in the shared and individual parameterization have been inferred from a ground-truth trajectory with $50\text{mm}\,\text{s}^{-1}$ knife velocity from a commercial solver (see subsubsection V-D2).

no two fruits or vegetables of the same type are identical. Since the mesh topologies can differ significantly between the different geometric shapes that a foodstuff may have, a direct mapping between the virtual nodes (respective cutting springs) is not possible, which would allow the transfer of the individual parameters. Instead, we propose a weighted mapping of a combination of cutting spring parameters from the source mesh that are in proximity to the cutting springs of the target mesh. We developed an optimal transport [70] method that receives as inputs the cutting spring coordinates in 2D (obtained by the mesh preprocessing step from subsection III-B) at the cutting interface (shown in Figure 12) from a source domain, and a different set of 2D coordinates for the target domain. By minimizing the Earth Mover's Distance (EMD) [74] between the cutting spring vertices of the two meshes[2], we find a weight matrix that allows us to compute spring parameters for the target domain as a weighted combination of the parameters from the source mesh. For more details, see subsection A-D. Similar to the velocity generalization experiments, we observe a significantly improved NMAE performance (Table I) in most cases when the cutting spring parameters are tuned individually ("NMAE OT") in contrast to duplicating average of each spring parameter ("NMAE Avg") from the source domain across all locations in the target domain.

Overall, our generalization experiments have shown that, although we optimize hundreds of parameters involved in the cutting dynamics at highly localized places, such representation still generalizes between various conditions. The successful transport of these parameters between two topologically different meshes based on their spatial correspondences indicates that our simulation parameters implicitly encode material properties that generalize across mesh topologies.

[2]Our implementation uses the Python Optimal Transport library [18]

Material	Source Mesh	Target Mesh	NMAE OT	NMAE Avg
Potato	Real Potato 1	Real Potato 2	0.948	5.635
Potato	Real Potato 2	Real Potato 1	1.360	6.981
Apple	Real Apple 2	Real Apple 3	6.844	1.330
Apple	Real Apple 3	Real Apple 2	3.857	19.749
Potato	Cylinder	Prism	3.470	12.001
Potato	Prism	Cylinder	0.933	1.983
Potato	Prism	Sphere	7.867	15.841
Potato	Sphere	Prism	35.347	16.812
Apple	Cylinder	Sphere	4.261	14.457
Apple	Sphere	Cylinder	1.839	1.602
Apple	Prism	Sphere	0.920	4.531
Apple	Sphere	Prism	13.883	45.983
Cucumber	Cylinder	Sphere	66.672	71.102
Cucumber	Sphere	Cylinder	4.239	0.484
Cucumber	Cylinder	Prism	58.138	64.407
Cucumber	Prism	Cylinder	1.046	1.855

Table I: Mesh generalization results when the parameters from the source domain are transferred to the target domain via Optimal Transport (OT), and by averaging the parameters across all cutting springs. The numbers show the normalized mean absolute error (NMAE), i.e., the MAE divided by the mean of the respective ground-truth knife force profile, to make the results comparable across different material properties and geometries.

E. Controlling Knife Velocity

In real-world applications, automated cutting of foodstuffs may need to meet multiple competing objectives. In particular, force may be minimized to prevent peripheral damage to the object, or ensure human safety, while the velocity may be maximized to reduce the required time. For cutting of food and biomaterials, humans have intuitively developed the strategy of minimizing cutting force by pressing the knife vertically and simultaneously slicing horizontally (i.e., a sawing motion) [3, 15, 16]. Such a cutting action can be intuitively understood as follows: by definition, the work applied by the knife to the object is equal to the cutting force integrated over displacement, and by conservation of energy, also equal to the fracture energy required to introduce a cut in the vertical plane. By simultaneously slicing horizontally, the distance traveled by the knife will be greater, reducing the cutting force for the required fracture energy.

Using gradient-based trajectory optimization, we observe that such an intuitive cutting strategy emerges. We define the cost function in Equation A4 to penalize the mean knife force and inverse velocity, and we parameterize the knife trajectory via keyframes in time that define the downward velocity and sinusoidal time-varying horizontal velocity (a complete description is given in section D of the appendix).

When optimizing Equation A4 through Adam without constraints on the sideways knife position, the resulting motion after 50 iterations (orange line in right subplot of Figure 13) consists of an initial pressing phase up to the point of contact with the cucumber, after which the knife continuously moves sideways without sawing. To limit the sideways motion to remain within the bounds of the blade length, we add an inequality constraint in Equation A5. By performing constrained optimization with the modified differential method of

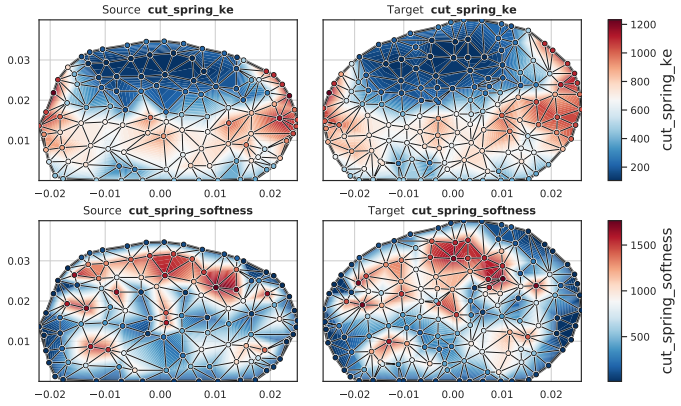

Fig. 12: Transfer of cutting parameters between two different potato meshes from a real-world cutting dataset. For the model in the left column, the cutting spring parameters have been optimized individually for each cutting spring given a single trajectory of the knife force (only two of the parameter types are shown in both rows). These parameters have been transferred to the mesh on the right column via Optimal Transport with the Earth Mover's Distance (EMD) objective (more details for this mesh transfer example are shown in Figure A2).

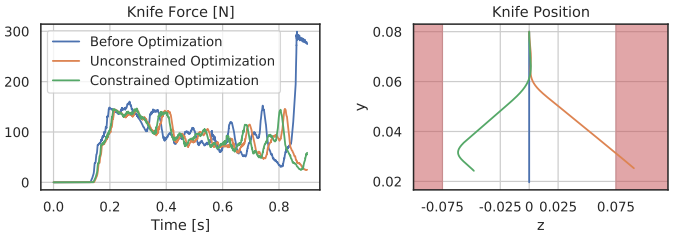

Fig. 13: Results from the trajectory optimization experiment (subsection V-E) on the cylinder mesh with cucumber material properties where the mean knife force is penalized and the overall downward velocity maximized. *Left:* knife force profiles before the trajectory optimization (blue), after unconstrained (orange) and constrained (green) optimization. *Right:* resulting knife motions (starting from $y = 8$ cm moving downwards), with constraints on the lateral motion shaded in red.

multipliers (MDMM) (see section D), we see that the knife moves within the bounds of the 15 cm blade length (green line on the right in Figure 13), while incurring only slightly more mean knife force (76.726 N) compared to the solution from the earlier unconstrained optimization (76.498 N). For comparison, the mean knife force was 89.698 N before the trajectory optimization.

VI. Conclusions

In this paper, we presented the first differentiable simulator for the robotic cutting of soft materials. Differentiability was achieved through a continuous contact formulation, the insertion of virtual nodes along a cutting plane, and a continuous damage model based on the progressive weakening of springs. The advantages of differentiability were shown through a systematic comparison of multiple gradient-based and derivative-free methods for optimizing the simulation parameters; leveraging gradients from the simulator enabled highly efficient estimation of posteriors over hundreds of parameters. The calibrated simulator was able to reproduce ground-truth data from a state-of-the-art commercial simulator in a fraction of the time, as well as closely match data from a real-world cutting dataset. Simulator predictions generalized across cutting velocities, object instances, and object geometries. Finally, a control experiment was performed in which human pressing-and-slicing behavior emerged from sample-efficient constrained optimization applied to the differentiable simulator, reducing the mean knife force by 15% relative to a vertical cut.

In future work, we plan to make multiple extensions. First, we will extend the material model for the soft material to explicitly capture nonlinearity and isotropy, as commonly observed in biomaterials; explicitly specifying such behaviors will facilitate optimization of simulator parameters. In addition, we will extend our modeling approach to accommodate more complex cutting actions, such as carving, in which the knife may arbitrarily rotate and follow more diverse trajectories than sawing motions; this approach will be used in additional control experiments, with the resulting trajectories compared again to human actions. Finally, we will evaluate calibrated simulator results against our own experimental dataset collected with an instrument robotic manipulator. Ultimately, we envision the use of differentiable cutting simulators in applications as challenging as robotic surgery, where tissue parameters can accurately and efficiently be estimated on initial contact with the cutting instrument, and the calibrated simulator can be used for faster-than-real-time roll-outs in a model-predictive framework for online surgical planning.

Acknowledgments

We thank Yan-Bin Jia and Prajjwal Jamdagni for providing the dataset of real-world cutting trajectories and meshes that we used in our experiments. We thank Krishna Mellachervu for outstanding technical support with commercial solvers. Finally, we thank Mike Skolones for his mentorship. This work was supported by a Google Ph.D. Fellowship.

References

[1] P Areias and Timon Rabczuk. Steiner-point free edge cutting of tetrahedral meshes with applications in fracture. *Finite Elements in Analysis and Design*, 132:27–41, 2017.

[2] A.G. Atkins and T. Atkins. *The Science and Engineering of Cutting: The Mechanics and Processes of Separating, Scratching and Puncturing Biomaterials, Metals and Non-metals*. Elsevier Science, 2009.

[3] AG Atkins, X Xu, and G Jeronimidis. Cutting by 'pressing and slicing' of thin floppy slices of materials illustrated by experiments on cheddar cheese and salami. *Journal of Materials Science*, 39:2761–2766, 2004.

[4] T Belytschko, YY Lu, and L Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[5] Ted Belytschko, Wing Kam Liu, Brian Moran, and Khalil Elkhodary. *Nonlinear finite elements for continua and structures*. John Wiley & Sons, 2013.

[6] I. Berndt, R. Torchelsen, and A. Maciel. Efficient surgical cutting with position-based dynamics. *IEEE Computer Graphics and Applications*, 37(3):24–31, 2017.

[7] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. In *11th Pacific Conference on Computer Graphics and Applications*, pages 377–386, 2003.

[8] Peter Brown. Contact modelling for forward dynamics of human motion. Master's thesis, University of Waterloo, 2017.

[9] Daniel Burkhart, Bernd Hamann, and Georg Umlauf. Adaptive and feature-preserving subdivision for high-quality tetrahedral meshes. *Computer Graphics Forum*, 29(1):117–127, 2010.

[10] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and Systems*, 2018.

[11] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[12] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

[13] W. H. Cunningham. A network simplex method. *Mathematical Programming*, 11:105–116, 1976.

[14] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems 31*, pages 7178–7189. 2018.

[15] Debao Zhou, M. R. Claffee, Kok-Meng Lee, and G. V. McMurray. Cutting, "by pressing and slicing", applied to robotic cutting bio-materials. i. modeling of stress distribution. In *Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 2896–2901, 2006.

[16] Debao Zhou, M. R. Claffee, Kok-Meng Lee, and G. V. McMurray. Cutting, 'by pressing and slicing', applied to the robotic cut of bio-materials. ii. force during slicing and pressing cuts. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2256–2261, 2006.

[17] IY El Said, Mervat M Atallah, KS Khalil, and AM El-Lithy. Physical and mechanical properties of cucumber applied to seed extractor. *Journal of Soil Sciences and Agricultural Engineering*, 2(8):871–880, 2011.

[18] R'emi Flamary and Nicolas Courty. Pot python optimal transport library, 2017. URL https://pythonot.github.io/.

[19] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. In *arXiv*, 2020.

[20] Markus Giftthaler, Michael Neunert, Markus Stäuble, Marco Frigerio, Claudio Semini, and Jonas Buchli. Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, 31(22):1225–1237, 2017.

[21] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, 12 1977.

[22] Andreas Griewank and Andrea Walther. Introduction to automatic differentiation. *PAMM*, 2(1):45–49, 2003.

[23] Alan Arnold Griffith. The phenomena of rupture and flow in solids. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 221(582-593):163–198, 1921.

[24] David Hahn, Pol Banzet, James M Bern, and Stelian Coros. Real2sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

[25] Eric Heiden, David Millard, Hejia Zhang, and Gaurav S. Sukhatme. Interactive differentiable simulation, 2020.

[26] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. NeuralSim: Augmenting differentiable simulators with neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://github.com/google-research/tiny-differentiable-simulator.

[27] Kelvin Hsu and Fabio Ramos. Bayesian learning of conditional kernel mean embeddings for automatic likelihood-free inference. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2631–2640, 16–18 Apr 2019.

[28] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4):60:1–60:14, July 2018. ISSN 0730-0301.

[29] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4): 150, 2018.

[30] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. ChainQueen: A real-time differentiable physical simulator for soft robotics. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[31] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[32] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[33] Michael Innes. Don't unroll adjoint: Differentiating ssa-form programs, 2019.

[34] Prajjwal Jamdagni and Yan-Bin Jia. Robotic cutting of solids based on fracture mechanics and fem. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 8246–8251, 2019.

[35] L. Jeřábková and T. Kuhlen. Stable cutting of deformable objects in virtual environments using XFEM. *IEEE Computer Graphics and Applications*, 29(2):61–71, 2009.

[36] Amir R Khoei. *Extended finite element method: theory and applications*. John Wiley & Sons, 2014.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations (ICLR)*, 2015.

[38] Twan Koolen and Robin Deits. Julia for robotics: simulation and real-time control in a high-level programming language. In *International Conference on Robotics and Automation*, 05 2019.

[39] Dan Koschier, Sebastian Lipponer, and Jan Bender. Adaptive tetrahedral meshes for brittle fracture simulation. *Symposium on Computer Animation*, 2014.

[40] Dan Koschier, Jan Bender, and Nils Thuerey. Robust extended finite elements for complex cutting of deformables. *ACM Trans. Graph.*, 36(4), July 2017.

[41] Sanjay Krishnan, Animesh Garg, Richard Liaw, Brijen Thananjeyan, Lauren Miller, Florian T Pokorny, and Ken Goldberg. SWIRL: A Sequential Windowed Inverse Reinforcement Learning Algorithm for Robot Tasks With Delayed Rewards. *International Journal of Robotics Research (IJRR)*, jul 2018.

[42] Ian Lenz, Ross A Knepper, and Ashutosh Saxena. DeepMPC: Learning deep latent features for model predictive control. *Robotics: Science and Systems*, 2015.

[43] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence

Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *AAAI*, 2016.

[44] Yancong Li, Xiaoyong Du, Jinhai Wang, and Chai Lei. Study on mechanical properties of apple picking damage. In *Proceedings of the 2017 3rd International Forum on Energy, Environment Science and Materials (IFEESM 2017)*, pages 1666–1670. Atlantis Press, 2018/02.

[45] MB Liu and GR Liu. Smoothed particle hydrodynamics (sph): An overview and recent developments. *Archives of computational methods in engineering*, 17:25–76, 2010.

[46] Lennart Ljung. *System Identification (2nd Ed.): Theory for the User*. Prentice Hall PTR, USA, 1999. ISBN 0136566952.

[47] Philip Long, Amine Moughlbay, Wisama Khalil, and Philippe Martinet. Robotic meat cutting. In *ICT-PAMM Workshop*, 2013.

[48] Rosa M. Sánchez-Banderas and Miguel A. Otaduy. Strain rate dissipation for elastic deformations. *Computer Graphics Forum*, 37(8):161–170, 2018.

[49] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Zach Corse. Local optimization for robust signed distance field collision. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(1), April 2020.

[50] Rolf Mahnken. Identification of material parameters for constitutive equations. *Encyclopedia of Computational Mechanics Second Edition*, pages 1–21, 2017.

[51] Charles C. Margossian. A review of automatic differentiation and its efficient implementation. *WIREs Data Mining and Knowledge Discovery*, 9(4), Mar 2019.

[52] C. Matl, Y. Narang, R. Bajcsy, F. Ramos, and D. Fox. Inferring the material properties of granular media for robotic tasks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2770–2777, 2020.

[53] Carolyn Matl, Yashraj S. Narang, Dieter Fox, Ruzena Bajcsy, and Fabio Ramos. STReSSD: Sim-to-real from sound for stochastic dynamics. *Conference on Robot Learning*, 2020.

[54] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher Pal, and Liam Paull. Active domain randomization. *Conference on Robot Learning*, 2019.

[55] Bhairav Mehta, Ankur Handa, Dieter Fox, and Fabio Ramos. A user's guide to calibrating robotics simulators. *Conference on Robot Learning*, 2020.

[56] M Eugene Merchant. Mechanics of the metal cutting process. i. orthogonal cutting and a type 2 chip. *Journal of applied physics*, 16(5):267–275, 1945.

[57] I. Mitsioni, Y. Karayiannidis, J. A. Stork, and D. Kragic. Data-driven model predictive control for the contact-rich task of food cutting. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 244–250, 2019.

[58] Nicolas Moës, John Dolbow, and Ted Belytschko. A finite element method for crack growth without remeshing. *International journal for numerical methods in engineering*, 46(1):131–150, 1999.

[59] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph.*, 23(3):385–392, August 2004.

[60] JJ Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30:543–574, 1992.

[61] Seyyed Javad Mousavizadeh, Ali N. Mashayekhi, Amir Daraei Garmakhany, Ab Ehtesham Nia, and Jafari M. Evaluation of some physical properties of cucumber (cucumis sativus l.). *Journal of Agricultural Science and Technology*, 4:107–115, 01 2010.

[62] Melissa Mozifian, Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek. Learning domain randomization distributions for training robust locomotion policies. *arXiv preprint arXiv:1906.00410*, 2019.

[63] Xiaoqian Mu, Yuechuan Xue, and Yan-Bin Jia. Robotic cutting: Mechanics and control of knife motion. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3066–3072, 2019.

[64] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[65] J. Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradSim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2021.

[66] Yashraj Narang, Balakumar Sundaralingam, Miles Macklin, Arsalan Mousavian, and Dieter Fox. Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections. *International Conference on Robotics and Automation*, 2021.

[67] Junjun Pan, Junxuan Bai, Xin Zhao, Aimin Hao, and Hong Qin. Real-time haptic manipulation and cutting of hybrid soft tissue models by extended position-based dynamics. *Computer Animation and Virtual Worlds*, 26(3-4):321–335, 2015.

[68] George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with Bayesian conditional density estimation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[69] Christoph J Paulus, Lionel Untereiner, Hadrien Courtecuisse, Stéphane Cotin, and David Cazier. Virtual cutting of deformable objects based on efficient topological operations. *The Visual Computer*, 31(6):831–841, 2015.

[70] G. Peyré and M. Cuturi. *Computational Optimal Transport: With Applications to Data Science*. Foundations and trends in machine learning. Now Publishers, 2019. ISBN 9781680835502.

[71] John C Platt and Alan H Barr. Constrained differential optimization for neural networks. *Advances in Neural Information Processing Systems*, 1988.

[72] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. Scalable differentiable physics for learning and control. In *ICML*, 2020.

[73] Fabio Ramos, Rafael Possas, and Dieter Fox. BayesSim: adaptive domain randomization via probabilistic inference for robotics simulators. In *Robotics: Science and Systems (RSS)*, 2019.

[74] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[75] Eftychios Sifakis, Kevin G Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 73–80, 2007.

[76] Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ronald Fedkiw. Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 81–90, 2007.

[77] Breannan Smith, Fernando De Goes, and Theodore Kim. Stable neo-hookean flesh simulation. *ACM Trans. Graph.*, 37(2), March 2018.

[78] Brijen Thananjeyan, Animesh Garg, Sanjay Krishnan, Carolyn Chen, Lauren Miller, and Ken Goldberg. Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning. In *IEEE International Conference on Robotics and Automation (ICRA)*, jun 2017.

[79] T. Tieleman and G. Hinton. Coursera: Neural netwroks for machine learning (lecture 6.5 - rmsproprop), 2012.

[80] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael P.H Stumpf. Approximate Bayesian computation

scheme for parameter inference and model selection in dynamical systems. *Journal of The Royal Society Interface*, 6(31): 187–202, Jul 2008.

[81] Agricultural Research Service U.S. Department of Agriculture. Fooddata central, 2019. URL fdc.nal.usda.gov.

[82] Stephanie Wang, Mengyuan Ding, Theodore F. Gast, Leyi Zhu, Steven Gagniere, Chenfanfu Jiang, and Joseph M. Teran. Simulation and visualization of ductile fracture with the material point method. *Proc. ACM Comput. Graph. Interact. Tech.*, 2(2), July 2019.

[83] Yuting Wang. *Virtual Node Algorithms for Simulating and Cutting Deformable Solids*. PhD thesis, UCLA, 2014.

[84] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *International Conference on Machine Learning (ICML)*, pages 681–688, 2011.

[85] L. Wijayarathne, Q. Sima, Z. Zhou, Y. Zhao, and F. L. Hammond. Simultaneous trajectory optimization and force control with soft contact mechanics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3164–3171, 2020.

[86] Joshuah Wolper, Yu Fang, Minchen Li, Jiecong Lu, Ming Gao, and Chenfanfu Jiang. Cd-mpm: Continuum damage material point methods for dynamic fracture animation. *ACM Trans. Graph.*, 38(4), 2019.

[87] Joshuah Wolper, Yunuo Chen, Minchen Li, Yu Fang, Ziyin Qu, Jiecong Lu, Meggie Cheng, and Chenfanfu Jiang. Anisompm: Animating anisotropic damage mechanics. *ACM Trans. Graph.*, 39(4), 2020.

[88] CT Wu, Y Guo, and W Hu. An introduction to the ls-dyna smoothed particle galerkin method for severe deformation and failure analyses in solids. In *International LS-DYNA Users Conference*, pages 1–20, 2014.

[89] C.T. Wu, Youcai Wu, John E. Crawford, and Joseph M. Magallanes. Three-dimensional concrete impact and penetration simulations using the smoothed particle Galerkin method. *International Journal of Impact Engineering*, 106:1–17, 2017.

[90] Jun Wu, Rüdiger Westermann, and Christian Dick. A survey of physically based simulation of cuts in deformable bodies. *Computer Graphics Forum*, 34(6):161–187, 2015.

APPENDIX A
ADDITIONAL ALGORITHMIC DETAILS

A. Adam Optimizer

At its core, Adam updates the parameters θ interactively as $\theta_i \leftarrow \theta_{i-1} - \alpha \cdot \hat{m}_i/(\sqrt{\hat{v}_i} + \varepsilon)$, where α is the learning rate, $\hat{m}_i$ and $\hat{v}_i$ represent the first and second order decaying averages (or momentum) after correcting for biases, and ε is a small value to prevent numerical issues. The full algorithm is shown in Algorithm 3. In Algorithm 4, we describe Stochastic Gradient Langevin Dynamics (SGLD) with the Adam update rule. A detailed description of SGLD is given in subsection IV-B.

B. Loss Functions

We investigated various loss functions for evaluating the closeness between knife force profiles effectively. Given the two-dimensional parameter inference experiment from subsection V-A, we test the following cost functions which we minimize via the Adam optimizer:

- $L1$ loss

Algorithm 2: Frank-Wolfe method

Result: Barycentric coordinate u of point on edge (p_1, p_2) with lowest signed distance w.r.t. SDF $d : \mathbb{R}^3 \to \mathbb{R}$

$u \leftarrow 1/2$
for $i = 0 \ldots$ max_iterations **do**
  $\delta \leftarrow \frac{\partial}{\partial u} d((1-u)p_1 + up_2)$
  if $\delta < 0$ **then**
    $s \leftarrow 1$
  else
    $s \leftarrow 0$
  end
  $\gamma \leftarrow \frac{2}{2+i}$
  $u \leftarrow u + \gamma(s-u)$
end

Algorithm 3: Stochastic gradient descent with Adam

Given: learning rate α, initial parameters θ_0, likelihood function $l(\cdot)$

$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
for $i = 1 \ldots$ max_iterations **do**
  $g_i \leftarrow \nabla_\theta l(\theta_{i-1})$
  $m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1) \cdot g_i$
  $v_i \leftarrow \beta_2 \cdot v_{i-1} + (1-\beta_2) \cdot g_i^2$
  $\hat{m}_i \leftarrow m_i/(1-\beta_1^i)$
  $\hat{v}_i \leftarrow v_i/(1-\beta_2^i)$
  $\theta_i \leftarrow \theta_{i-1} - \alpha \cdot \hat{m}_i/(\sqrt{\hat{v}_i} + \varepsilon)$
end
return θ_i

Algorithm 4: SGLD with Adam update rule

Given: learning rate α, initial parameters θ_0, likelihood function $l(\cdot)$

$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
for $i = 1 \ldots$ max_iterations **do**
  $g_i \leftarrow \nabla_\theta U(\theta_{i-1})$
  $m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1) \cdot g_i$
  $v_i \leftarrow \beta_2 \cdot v_{i-1} + (1-\beta_2) \cdot g_i^2$
  $\hat{m}_i \leftarrow m_i/(1-\beta_1^i)$
  $\hat{v}_i \leftarrow v_i/(1-\beta_2^i)$
  $A_i \leftarrow \mathrm{diag}\left(\mathbf{1} \oslash (\sqrt{\hat{v}_i} + \varepsilon)\right)$
  $\eta_i \sim \mathcal{N}(0, \alpha A_i)$
  $\theta_i \leftarrow \theta_{i-1} - \alpha \cdot \hat{m}_i \cdot A_i + \eta_i$
end
return θ_i

- $L2$ loss
- Inverse Cosine Similarity
- LogSumExp

We visualize the trace of the optimized parameters in Figure A1. The $L1$ loss function compares favorably to the other formulations as it allows for a relatively fast convergence to the true parameters, while also yielding a "bouncing" behavior when the iterates are close to the (local) optimum. Since we use the $L1$ loss function in the likelihood term for our probabilistic parameter inference experiments V-A, the likelihood distribution corresponds to a Laplace distribution which is known to have sharp peaks and heavy tales.

C. BayesSim Implementation

BayesSim approximates the posterior over the simulation parameters as

$$p(\theta|\phi) \approx p(\theta)/\tilde{p}(\theta)\, q(\theta|\phi),$$

where $p(\theta)$ is the prior distribution and $\tilde{p}(\theta)$ is a proposal prior used to sample the simulator and collect N samples, $\{\theta_i, \phi_i^s\}_{i=1}^N$, to learn $q(\theta|\phi)$. When a real trajectory ϕ^r is observed, BayesSim computes $p(\theta|\phi = \phi^r)$ that represents the posterior over simulation parameters given the real data. As inferring simulator parameters given trajectories is a type of inverse problem, it can admit a multitude of solutions.

For the BayesSim baseline, we train a mixture density network (MDN) representing $q(\theta|\phi)$ with 10 components on a dataset consisting of 500 knife force trajectories that have been generated in our simulator by uniformly sampling the simulation parameters within their respective bounds. Depending on the experiment, we limit ourselves to only a subset of all the available parameters, due to the exponential increase in sample complexity. As summary statistics input to BayesSim, we downsample the 0.9 s knife force profiles consisting of 90,000 time steps by a factor of 1000 to 90-dimensional summary statistics using polyphase filtering. While training the MDN, we project the true parameter values to the unit interval using their possible ranges (see Table A2), as we found the MDN

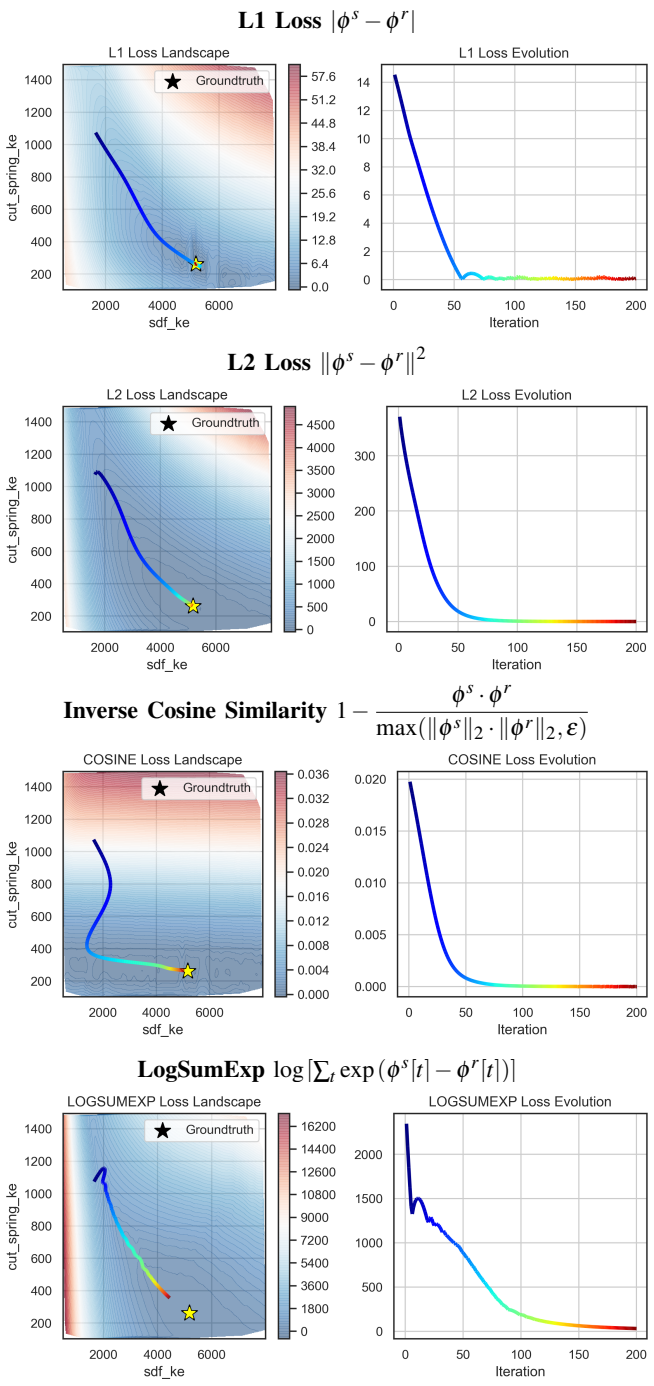

Fig. A1: Evaluation of various loss functions on the `actual_2d` scenario from subsection V-A. For 500 randomly sampled parameter vectors, the trajectories are simulated (shown as dots) and the loss is computed between the simulated trajectory ϕ^s and the ground-truth knife force profile ϕ^r. The interpolated loss landscape is shown one the left, and the training loss evolution using the Adam optimizer is shown on the right. Overall, the $L1$ loss performs favorably compared to the other cost functions and is therefore our choice to evaluate the closeness between knife force trajectories across this work.

Material	E ($\mathrm{N\,m^{-2}}$)	ν	ρ ($\mathrm{kg\,m^{-3}}$)
Apple	3.0×10^6	0.17	787
Potato	2.0×10^6	0.45	630
Cucumber	2.5×10^6	0.37	950

Table A1: Properties of common biomaterials: Young's modulus E, Poisson's ratio ν, density ρ.

to be sensitive to the scale of the parameters, and to perform more accurately with homogeneous value ranges.

D. Optimal Transport of Cutting Spring Parameters

The Earth Mover's Distance can be interpreted as the minimal cost associated with transforming a constant volume pile of dirt into another, where the cost is defined as the amount of dirt moved multiplied by the distance travelled. Formally, given two sets of vertices P and Q, associated sets of vertex weights $\mathbf{w}_P$ and $\mathbf{w}_Q$, and a cost matrix given by the Euclidean distance between two points $D = |d_{i,j}| = \|p_i - q_j\|^2$, optimal transport finds the solution of the optimization problem

$$\min_F \frac{1}{Z} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} d_{i,j} \tag{A1}$$

where $F = f_{i,j}$ is the movement or flow between p_i and q_j which we try to minimize, and $Z = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j}$ is a normalization constant. In our formulation, we assume uniform weights for both sets of vertices.

This problem can be efficiently solved using the network simplex algorithm [13] and has typical complexity $O(n^3)$, but sparsity can be exploited to reduce this cost.

Appendix B
Details of Experimental Setup

A. Simulation Parameters

DiSECt introduces additional degrees of freedom via the insertion of virtual nodes and cutting springs that connect them (see subsection III-B). In Table A2, we list each available parameter with its description and default value. Parameters that are allowed to be individually tuned for each spring can be set in two modes:

- Shared parameterization: the parameter is a single scalar that gets replicated across all cutting springs.
- Individual parameterization: the parameter is a vector where each entry can be tuned separately for each spring.

To enforce hard parameter limits in our simulator, throughout our experiments, we impose bounds on the estimated simulation parameters by projecting them through the sigmoid function. Thus, given an unconstrained real number x to be optimized, the resulting projected parameter value is $\mathrm{sigmoid}(x) \cdot (p_{ub} - p_{lb}) + p_{lb}$, where p_{lb} and p_{ub} are the upper and lower bounds of the parameter, and $\mathrm{sigmoid}(x) = 1/(1+\exp(-x))$.

Name	Description	Default value
Knife geometry parameters (fixed)		
edge_dim	Lower diameter of knife (see Figure A3 right)	0.08 mm
spine_dim	Upper diameter of knife (spine)	2 mm
spine_height	Height of knife spine	40 mm
tip_height	Height of knife tip	0.04 mm
depth	Length of knife blade (along z axis)	150 mm
Knife motion		
velocity_y	Vertical knife velocity	-0.05 $\mathrm{m\,s^{-1}}$
initial_y	Initial vertical knife position (height)	80 mm
Spring-damper parameters (individual for each spring)		
cut_spring_ke	Spring stiffness coefficient at the start of the simulation (initial stiffness of the cutting spring)	500
cut_spring_kd	Spring damping coefficient	0.1
cut_spring_softness	Softness coefficient γ used in the linear spring loosening (Equation 2)	500
Knife contact dynamics parameters (individual for each spring)		
sdf_radius	Radius around SDF to consider for contact dynamics	0.5 mm
sdf_ke	Positional penalty coefficient (contact normal stiffness)	1000
sdf_kd	Damping coefficient	1
sdf_kf	Contact friction stiffness (tangential stiffness used in Coulomb friction model)	0.01
sdf_mu	Friction coefficient (μ)	0.5
Ground contact dynamics parameters (fixed)		
ground_ke	Positional penalty coefficient (contact normal stiffness)	100
ground_kd	Damping coefficient	0.1
ground_kf	Contact friction stiffness (Coulomb friction model)	0.2
ground_mu	Friction coefficient (μ)	0.6
ground_radius	Radius around mesh vertices	1 mm
Material properties		
young	Young's modulus E	see Table A1
poisson	Poisson's ratio ν	see Table A1
density	Density ρ	see Table A1

Table A2: Overview of the model parameters used by our cutting simulator.

B. Commercial Simulation Setup

In the commercial solver, each simulation consisted of a rigid knife, 1 of 3 deformable fruits/vegetables (apple, cucumber, or potato), and a rigid table. The apple, cucumber, and potato geometries were represented by sphere, cylinder, and rectangular shape primitives, respectively. Each primitive had a 10 mm-thick slice, centered along the long axis, that defined the volumetric region that could be cut. All regions were assigned an isotropic elastic failure material model, with density, Lamé parameters (see Table A1), yield stress, and failure strain obtained from the agricultural mechanics literature [17, 44, 61] and the FoodData Central database from the U.S. Department of Agriculture [81]. The non-slice regions were simulated using FEM with a tetrahedral mesh-based discretization. The slice region was simulated using the smoothed particle Galerkin (SPG) method with a particle-based discretization. SPG is a numerical method related to smoothed particle hydrodynamics (SPH) [45, 60] and the element-free Galerkin method (EFG) [4], and has been validated for simulating large deformation and failure of elastoplastic solids [88]. Continuity conditions were imposed between the non-slice mesh and the slice particles. Coulomb frictional contact was defined between the knife and the deformable object, as well as between the object and the table, with a coefficient of friction of 0.6. A constant downward velocity was applied to the knife until contact with the surface of the table, and gravity was applied to the deformable object.

APPENDIX C
ADDITIONAL EXPERIMENTS

A. Generalization Results

Figure A4 shows the trajectories for the generalization experiment (subsubsection V-D1) where the parameters have been optimized for a shorter simulation time window of 0.4 s, and tested against the ground-truth trajectory (green, dashed line) over a duration of 0.9 s.

Analogous to the apple cutting results with ground-truth from the commercial solver in Figure 10, the bar plot in Figure A5 visualizes the normalized mean absolute error (NMAE) for different knife velocities at test time. The simulation parameters have been optimized for a vertical downward velocity of 50 mm (green shade in the background), given a knife force profile of cutting a cylindrical mesh with cucumber material properties from the commercial simulator. At test time, we use the same optimized simulation parameters to evaluate the accuracy of the force profile simulation against the commercial simulator on different downward velocities. As in the experiments of cutting an apple (Figure 10), the results in Figure A5 show that the individual cutting spring parameterization outperforms the shared parameterization across all tested velocities. While the performance improvement over the shared parameterization becomes less significant, it can

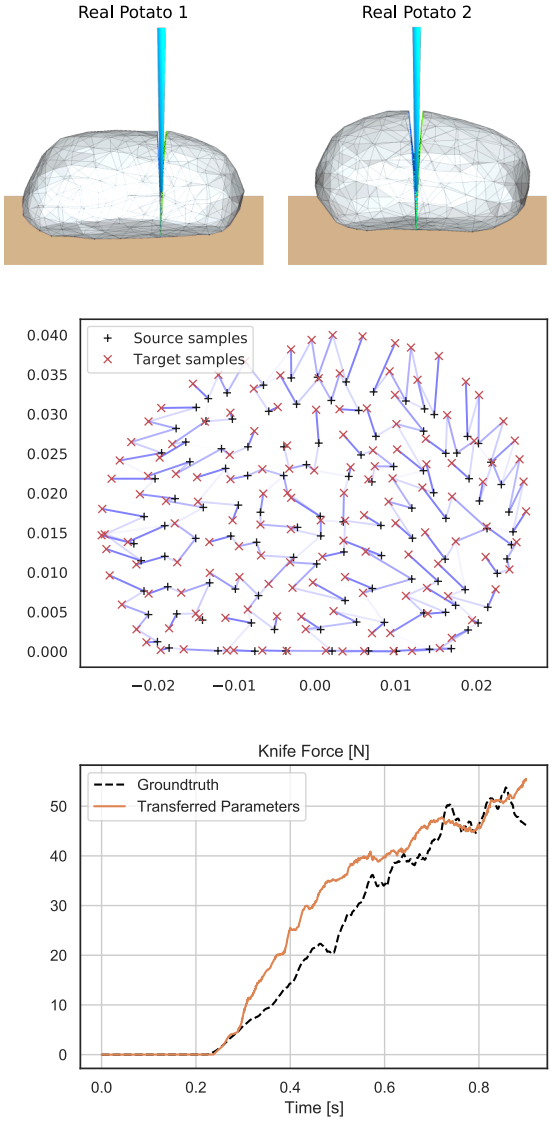

Fig. A2: Optimal transport of simulation parameters from one mesh of a real potato (top left) to another real potato mesh (top right). The correspondences (center) have been found via optimal transport using the Earth Mover's Distance (Equation A1) between cutting springs in two potato meshes. The weighted mapping between the 2D positions of the springs at the cutting interface from the source domain (`ybj_potato1`) to the target domain (`ybj_potato2`) is used to transfer the cutting spring parameters between the two topologically different meshes, resulting in a close fit to the ground-truth trajectory (bottom).

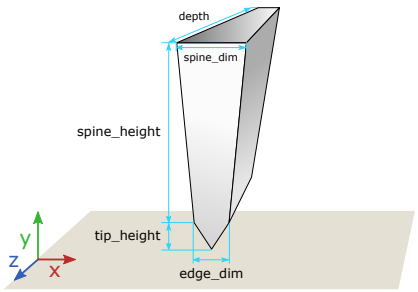

Fig. A3: Parameters that describe the knife geometry.

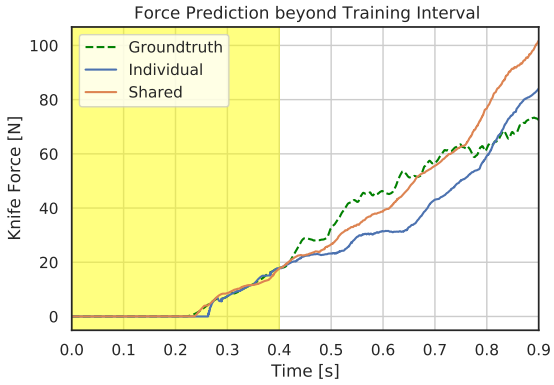

Fig. A4: Generalization performance on the apple cutting trajectory by simulating the parameters which have been optimized from a 0.4 s ground-truth trajectory (highlighted in yellow) from a commercial simulator. At test time, the force profile is simulated over a duration of 0.9 s (see subsubsection V-D1). The blue line shows the estimated trajectory when all parameters were optimized individually for each cutting spring. Shown in orange is the resulting trajectory from optimizing parameters shared across all cutting springs.

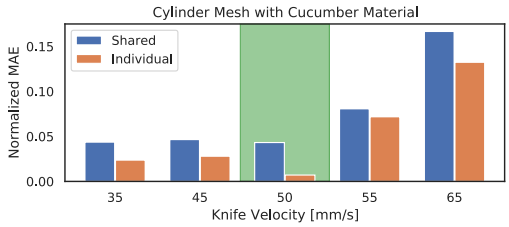

Fig. A5: Velocity generalization results for a cylinder mesh with cucumber material properties given ground-truth simulations with different vertical knife velocities from a commercial solver. Two versions of DiSECt were calibrated: by sharing the parameters across all cutting springs (blue) and by optimizing each value individually (orange), given a ground-truth trajectory with the knife sliding down at $50\,\mathrm{mm\,s^{-1}}$ speed (highlighted in green). The normalized mean absolute error (MAE) is evaluated against the ground-truth by rolling out the estimated parameters for the given knife velocity.

be concluded that the individual parameterization generalizes better to novel test velocities.

B. Posterior Over Simulation Parameters

The marginal plots in Figure A6 and Figure A7 show the posteriors from iterative BayesSim and SGLD, respectively. These densities over a subset of the simulation parameters (shared across the cutting springs) have been inferred from knife force profiles of cutting a sphere mesh using apple material properties. The ground-truth, from which the posteriors are inferred, stems from a high-fidelity commercial simulator (see description in subsection V-B).

Appendix D
Knife Motion Trajectory Optimization

We represent the knife velocity trajectory by k equidistant keyframes in time. Three parameters are to be optimized

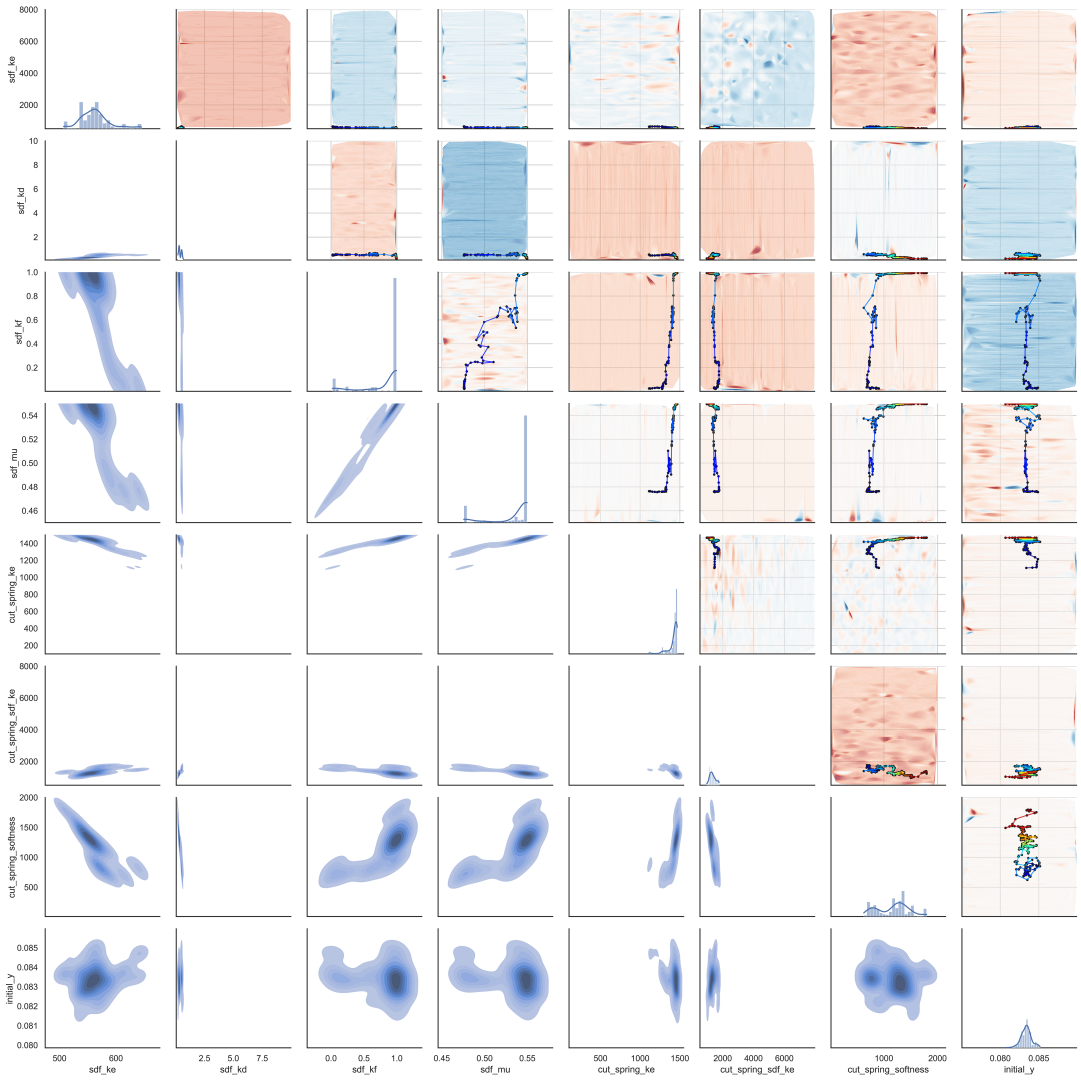

Fig. A6: Posterior obtained by SGLD in our differentiable simulator after 300 trajectory roll-outs. Marginals are shown on the diagonal. The sampled chain is visualized for pairs of parameter dimensions in the upper triangle, along with an approximate rendering of the loss surface as heatmap in the background. The heatmaps in the lower triangle visualize the kernel densities for all pair-wise combinations of the parameters.

per keyframe i (refer to Figure A3 for the coordinate frame orientation w.r.t. the knife):

- a_i: the amplitude of the lateral (along z axis) sinusoidal velocity
- b_i: the frequency of the lateral sinusoidal velocity
- c_i: the vertical (along y axis) velocity

To allow for a smooth interpolation between the keyframes, and propagation of gradients from all trajectory parameters at every time step, we weight the contribution of all keyframe parameters on the entire trajectory via the radial basis function (RBF) kernel. The kernel uses the squared norm of the difference between the current time t difference and the predefined keyframe times to compute the weight contributions $\mathbf{w} \in \mathbb{R}^k$ of the keyframe parameters (see Figure A8):

$$\mathbf{w}(t) = \exp\left(-\frac{\|t-w\|^2}{2\sigma^2}\right) \tag{A2}$$

In effect, a nonzero contribution on the trajectory is maintained from all keyframes at all times, which eases gradient-based optimization.

The kernel width σ controls how smoothed out the contributions of the keyframe parameters become. We found $\sigma = \sqrt{0.03}$ to be an appropriate setting (see illustration in Figure A8), given that the duration of the cutting action we optimize for is 0.9 s using $k = 5$ keyframes.

To compute the knife's horizontal (sideways) and vertical velocities $\dot{z}_{\text{knife}}(t)$ and $\dot{y}_{\text{knife}}(t)$ at time $t = [0..T]$, the keyframe

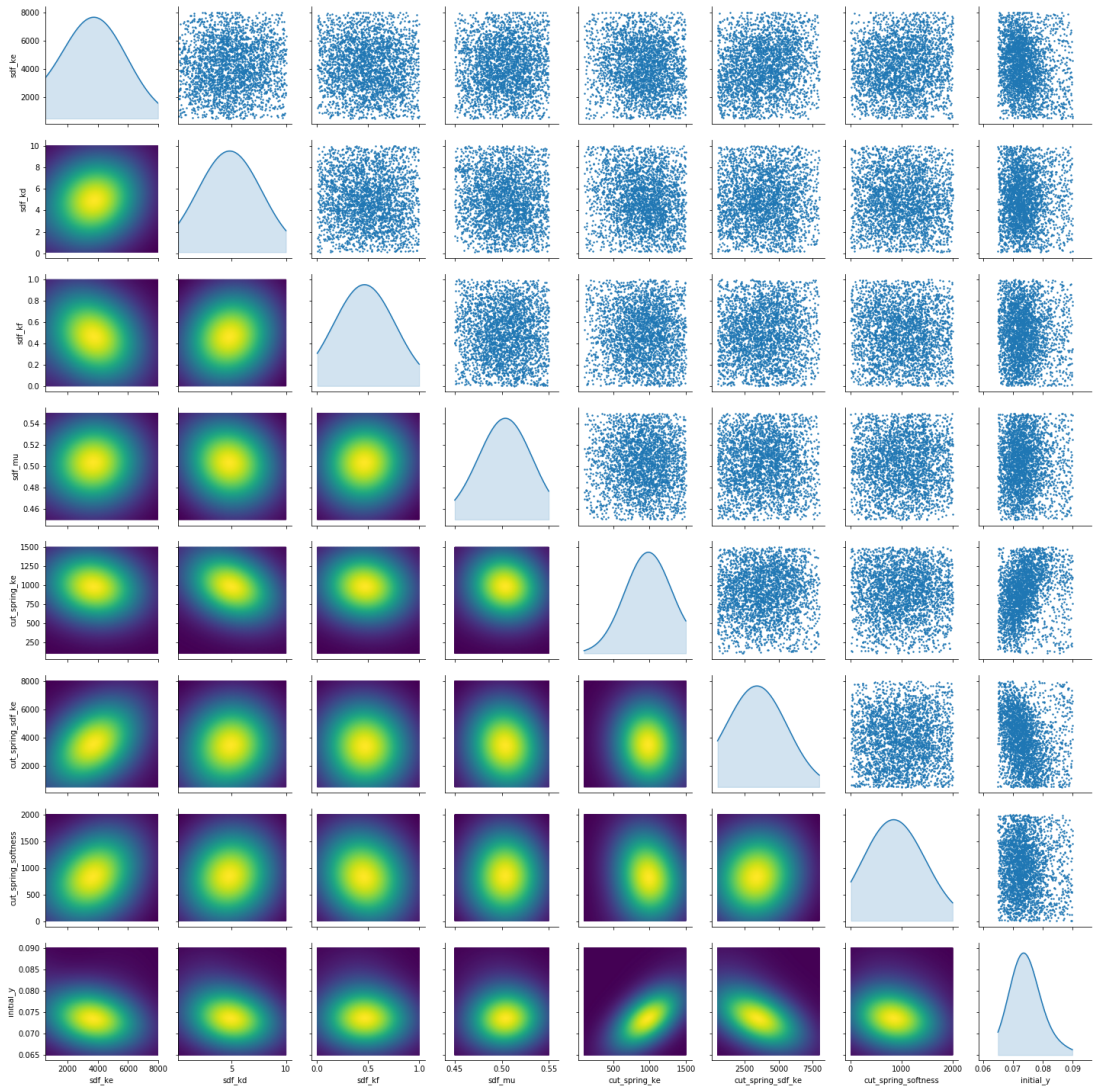

Fig. A7: Posterior obtained by BayesSim after 100 iterative updates of the training dataset with 20 new trajectories per iteration. Marginals are shown on the diagonal. Parameter samples are represented by the dots in the scatter plots from the upper triangle. The heatmaps in the lower triangle visualize the kernel densities for all pair-wise combinations of the parameters.

parameters in vector form $\mathbf{a},\mathbf{b},\mathbf{c} \in \mathbb{R}^k$ are combined with the time-dependent weighting contribution from Equation A2:

$$\dot{z}_{\text{knife}}(t) = \mathbf{a}\cdot\mathbf{w}(t)\cos(\mathbf{b}\cdot\mathbf{w}(t)t) \qquad \dot{y}_{\text{knife}}(t) = \mathbf{c}\cdot\mathbf{w}(t) \quad \text{(A3)}$$

We minimize the mean knife force plus the vertical knife velocity integrated over the entire length of the trajectory. Thus, we penalize high actuation effort by the robot controlling the knife while maintaining a fast progression of the cutting process:

$$\underset{\mathbf{u}=[\mathbf{a},\mathbf{b},\mathbf{c}]}{\text{minimize}} \qquad \mathfrak{L} = \frac{1}{T}\int f(t,\mathbf{a},\mathbf{b},\mathbf{c}) + \dot{y}_{\text{knife}}(t)\,dt \qquad \text{(A4)}$$

$$|z_{\text{knife}}(t)| \leq \frac{1}{2} l_{\text{knife}}, \qquad \text{(A5)}$$

where $k = 5$ is the number of keyframes, $T = 0.9\,\text{s}$ is the time at the end of the trajectory, f is the simulation step that returns the knife force norm $\|\mathbf{f}_{\text{knife}}\|$ at time step t given the trajectory parameter vectors $\mathbf{a},\mathbf{b},\mathbf{c}$, and $l_{\text{knife}} = 15\,\text{cm}$ is the blade length of the knife. We impose the hard constraint in Equation A5 to ensure that the knife does not move too far along z where the blade of the knife ends (which would trivially minimize the knife force, although we assume the knife blade has infinite length in this experiment setup).

Constrained optimization problems are typically solved by converting them to unconstrained optimization problems through the introduction of Lagrange multipliers. However, the critical points to such Lagrangians often tend to be saddle points, which gradient-descent-style algorithms, such as Adam, will not converge to [71]. To make the unconstrained objective amenable to gradient descent, following the modified differential method of multipliers (MDMM) [71], we introduce a penalty term for $\mathbf{u} = [\mathbf{a},\mathbf{b},\mathbf{c}]$:

$$E_{\text{penalty}} = \frac{c}{2}(g(\mathbf{u}))^2.$$

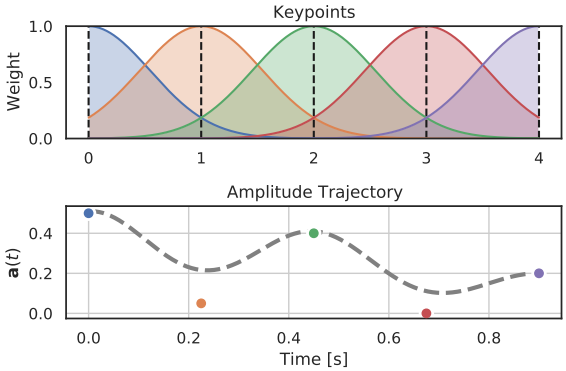

Fig. A8: Visualization of five keypoints (top) evenly distributed in time with exemplary amplitude values $\mathbf{a}[i]$ per keyframe i (color-matching dots in lower plot). The resulting continuous trajectory $\mathbf{a}(t)$ resulting from weighting the keyframes via the RBF kernel (Equation A2) is shown as the dashed line at the bottom.

This term acts as an attractor to the energy function that we are optimizing for (where c acts as a damping factor), where $g(\mathbf{u})$ is an equality constraint.

To include the inequality constraint in Equation A5, we convert it to an equality constraint $g(\mathbf{x})$ by introducing slack variable $\gamma \in \mathbb{R}$ that becomes part of $\mathbf{u}$:

$$g(\mathbf{u}) = \frac{1}{2} l_{\text{knife}} - |z_{\text{knife}}(t)| - \gamma^2. \tag{A6}$$

The update rule for the trajectory parameters $\mathbf{u}$ is then

$$\mathbf{u}' = \mathbf{u} - \frac{\partial \mathfrak{L}}{\partial \mathbf{u}} - \lambda \frac{\partial g}{\partial \mathbf{u}} - c g(\mathbf{u}) \frac{\partial g}{\partial \mathbf{u}} \tag{A7}$$

$$\lambda' = \lambda + g(\mathbf{u}). \tag{A8}$$