DOI: 10.1111/cgf.14720

COMPUTER GRAPHICS *forum*
Volume 42 (2023), number 1 pp. 277–289

Differentiable Depth for Real2Sim Calibration of Soft Body Simulations

K. Arnavaz,[1] M. Kragballe Nielsen,[1] P. G. Kry,[2] M. Macklin[3] and K. Erleben[1]

[1]University of Copenhagen, Copenhagen, Denmark
[2]McGill University, Montreal, Canada
[3]NVIDIA, Santa Clara, New Zealand

Abstract

In this work, we present a novel approach for calibrating material model parameters for soft body simulations using real data. We use a fully differentiable pipeline, combining a differentiable soft body simulator and differentiable depth rendering, which permits fast gradient-based optimizations. Our method requires no data pre-processing, and minimal experimental set-up, as we directly minimize the L2-norm between raw LIDAR scans and rendered simulation states. In essence, we provide the first marker-free approach for calibrating a soft-body simulator to match observed real-world deformations. Our approach is inexpensive as it solely requires a consumer-level LIDAR sensor compared to acquiring a professional marker-based motion capture system. We investigate the effects of different material parameterizations and evaluate convergence for parameter optimization in both single and multi-material scenarios of varying complexity. Finally, we show that our set-up can be extended to optimize for dynamic behaviour as well.

Keywords: animation, physically based animation, methods and applications, robotics, rendering, ray tracing

CCS Concepts: • Computing methodologies → Rendering; Physical simulation

1. Introduction

Selecting correct parameter values in material models is important in a wide range of areas such as modelling soft tissue [BT07], cloth [BTH*03, WOR11, MBT*12, MTB*13, YLL17, SNW20], fabrication [BKS*12, STC*13, CLSM15, CLMK17, YLYW18, WWY*15, WDK*20], haptics [VCO20, PKLD20] and robotics [YL16, BCC17, BBPC19, BSB*20, HBBC19, SLK*20]. Our work is inspired by the problem of producing a computer model of a soft robot that visually matches reality. In the field of computer graphics, many works have explored the field of material modelling to achieve more realistic-looking animations [XSZB15, XB17, XLCB15, BJ05, LB15, MTGG11, SGK18, ITF04, WZB17]. In our work, we use existing material models as these have been shown to work well for the soft robotics cases we present. We use CAD models directly in our work as we target soft fabricated objects, so there is no need for scanning geometry. We match the simulation to the real world without any need for complicated force measurements. As our approach is markerless, there is no need for expensive motion tracking equipment, nor do we need to deal with computer vision problems which are common in marker-based methods.

Our work is no different from any other approach for calibration of parameters in the sense that we are subject to the same fundamental challenges and pitfalls, i.e. changing boundary conditions or different deformation modes always lead to different estimated values. This is no surprise as data only shows one mode of deformation, and we optimize for that specific motion. This is not a drawback when using simulation for predicting real-time motions for control problems, where getting accurately simulated deformation is of crucial importance. The same requirement applies when solving design problems. Hence, the true strength of our work is that we offer a complete alternative technology that is far less sensitive to the challenges of image capture or difficulties with markers. We have created a simple yet versatile and robust approach for calibrating a simulator to the real world.

Theoretically, to estimate material parameters, one needs many pieces of information: the boundary conditions, force measure-

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd. This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

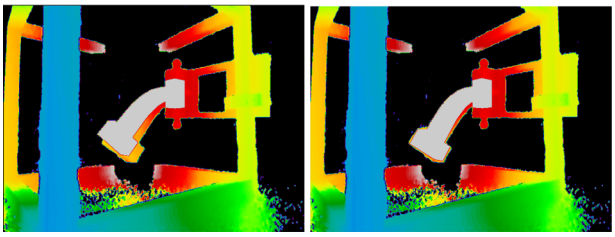


Figure 1: *Simulations with elastic parameters reported by silicone manufacturers (left) and fitted values (right). The depth camera image is shown in colour and the simulated objects are in grey. For Ecoflex-50, the nominal Young's modulus is 83 kPa, which does not match the real world when used with a coarse mesh simulation, while the value of 65 kPa is calculated by our method for an optimal match.*

ments, knowledge of the material model, the undeformed shape, a sufficiently high mesh resolution so that numerical stiffness is negligible and a large enough set of motion samples to capture sufficient variations of possible deformations. Expecting such complete information is unrealistic when working in the real world. For instance, one never knows the undeformed coordinates because gravity always creates a bias. Objects have imperfections, thus, no true homogeneous solid exists. Running simulations at high resolutions are impractical due to memory limitations. This last point has pushed us to use a coarse mesh to match reality with our simulator, which undoubtedly results in different elastic values than those measured in engineering laboratory settings. Similar to Hahn *et al.* [HBBC19], we argue that plugging engineering measured parameters into a coarse mesh simulator will fail to predict the real world. Figure 1 highlights this phenomenon with a didactic example. Consequently, our focus in this paper has been more on reducing the reality gap rather than, for instance, reaching the same elastic parameters listed on data sheets as reported by silicone manufacturers.

The key ingredient in our work is to combine differential depth rendering with differential physics and depth sensors. We create a simple system that uses the chain rule to provide gradient information directly from images, which can be easily integrated within existing optimization frameworks.

Differential rendering is related to inverse rendering and is an active research field [ZJL20]. Approaches range from finite difference-based approximate differential rendering [LB14], to ray tracing [LADL18] and scan conversion [CLG*19]. In this work, we apply a differentiable ray-tracer to emulate a LIDAR scanner.

The idea of using differentiable simulators for tuning parameters is obvious. Many works estimate mass and friction of rigid bodies [dABPSA*18, DHDw19, LLKL*20, KAMS20, SB20, MMG*21]. Most of these use synthetic data generated with simulators such as Bullet or MuJoCo. Others tune material parameters such as stiffness and density for cloth and soft bodies [LLK19, LL20, GHZ*20, MMG*21]. As demonstrated in Chen *et al.* [CLMK17], simulated behaviour depends heavily on the discretization of the soft object. Thus, parameters need to be re-calibrated whenever the object, its discretization or the simulator changes. Our work aims to create a robust and efficient pipeline, which can significantly reduce time spent setting up data capture and performing parameter tuning. Our contributions in this work include the following:

- A robust marker-less approach to compare real and simulated data;
- A straightforward way to perform gradient-based optimization of simulator parameters based on image similarity using a fully differential pipeline;
- Multiple real-world examples of both homogeneous and heterogeneous soft bodies.

The code and the data including 3D models, images and videos are publicly available at https://github.com/diku-dk/DiffCal.

2. Previous Work

Many works apply stereo-vision using multiple cameras in combination with markers or feature point detection to track deformations of material samples [BBO*09, WOR11, BKS*12, CLMK17, LLK19, HBBC19, LLKL*20, SB20, GHZ*20, LL20]. In some cases, texture and optical flow are used to track a surface grid during a controlled manipulation setup [MBT*12]. For cloth, there exists single camera setups that exploit the 2D nature of a cloth sample to create a simple way to measure cloth deformations [BTH*03, WOR11]. Marker-less methods exist and often use a physics-based probabilistic approach to track the deformation of the soft object. This is quite robust to noise and occlusion [YL16, WWY*15, WDK*20].

Our approach has no need for tracking markers or creating kinematic trajectories of known material positions. Instead, our method minimizes the difference between rendered and captured depth images. Control of light conditions, colours and textures is important for many works using stereo vision or cameras. Our work is less-dependent on such conditions as we only need depth information that can be captured by an inexpensive consumer-level LIDAR camera. Other works have also used depth sensors such as Kinect or Intel RealSense [WWY*15, GHZ*20, WDK*20] to capture depth data. In these works, a point-correspondence between the point cloud and a virtual model is created. This is used to determine the current deformation of the object. We have no need for establishing point correspondences, as we compare depth data directly. These works typically need multiple cameras, as they are sensitive to occlusion events. We show that by selecting camera angles carefully, we can reproduce material behaviour from one camera.

Many works have explored optimizing simulation parameters by minimizing the squared difference between tracked and simulated positions [WWY*15, dABPSA*18, HBBC19, BBPC19, LLK19, DHDw19, LLKL*20, KAMS20, GHZ*20, MMG*21]. Force equilibrium constraints can be added to the minimization problem when quasi-static problems are solved. This allows sensitivity analysis to be applied to determine gradients for the objective function [MBT*12, BCC17, YLYW18, HBBC19, BBPC19]. If the simulated positions are known as a function of material parameters, one may omit the force-equilibrium constraint [YLYW18].

Optimization methods that have been applied to find optimal simulation parameters fall into two categories. One is gradient-free approaches like the Nelder–Mead algorithm [WWY*15] and

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

CMA-ES [DHDw19]. The other category is based on gradient information and cover methods such as BFGS [WOR11, HBBC19], Levenberg–Marquardt [SLK*20], simulated annealing [BTH*03], descent methods [YLYW18] and similar. Gradient-free methods are inferior in terms of performance compared to gradient-based approaches [MMG*21]. Hence, our work uses a gradient-based approach. There is a sub-class of techniques that estimate gradients using finite differences [WOR11], Gaussian processes [PKLD20] and sampling [SLK*20]. However, these scale poorly and instead, we use a differentiable simulator to obtain exact gradients.

Recently, many works have presented differentiable simulators for rigid bodies, articulated rigid bodies, cloth and fluids [dABPSA*18, SF18, LLK19, DHDw19, LL20, LLKL*20, KAMS20, SB20, HMZS20]. Recent work addressed multi-body simulations of both rigid and soft bodies [MEM*20, GHZ*20]. In our work, we focus on hyper-elastic soft materials that are connected to rigid fixed objects. Our approach is most similar to that of Murthy *et al.* [MMG*21]. The techniques used for providing gradient information in differentiable simulators are many. Analytical gradients can be computed for linear complementary problem, quadratic programming models and staggered projections [dABPSA*18, LLK19, LLKL*20, LL20, KAMS20, SB20]. Many works rely on reverse mode auto differentiation [DHDw19, HMZS20, WAB20]. Recent work combines the Adjoint method, source code transformation and taping [GHZ*20, MMG*21]. A programming language for creating differentiable simulators has even been proposed by Refs. [HLS*19, HAL*20].

Wang *et al.* [WOR11] and Miguel *et al.* [MBT*12] both address cloth. In these works, a piece of square cloth is clamped on both sides and subjected to buckling manipulation. Results show examples comparing the buckling of a piece of cloth side-by-side with a real capture. Wang *et al.* [WOR11] show impressive complex draping examples proving that their optimized parameters generalize to simulations with complex geometry. It is difficult to assess how such complex draping compares to real ground truth data. In contrast, in our work, we focus on elastic solids and calibrate our simulations to both simple beam-like geometries and more complex shapes, such as soft robotic fingers and the XYZ Dragon model. In all our cases, we include a quantitative comparison to real data including a study of the variance. Furthermore, we use a relatively small number of Dirichlet conditions. Comparing over-constrained cloth capture to solid capture is not within the scope of our work because we instead target soft bodies in the context of soft robotics, where soft deformable fingers of various designs are often mounted on a fixed base and made to move using cables or pneumatic activation. We include an example showing a cable-driven soft-robot finger in our results.

Chen *et al.* [CLMK17] focus on elastic solids and their comparisons to the real world include optimized geometric designs which improve a user's ability to produce the desired motion, such as having a small jumping mechanism leap over an obstacle. Fast optimizations are aided by coarse models with tuned mechanical parameters so as to avoid numerical stiffening. Our work addresses numerical stiffening in coarse models in a similar manner.

Geilinger *et al.* [GHZ*20] are close to our work. The differences are that we use semi-implicit Euler and support heterogeneous materials without the use of markers. Murthy *et al.* [MMG*21] were the first to suggest using differentiable rendering in combination with a differentiable simulator, gradSim. They use a loss function on framebuffer pixels from synthetic examples only. To extend to real data, this method will need to address colours, light conditions and textures. We demonstrate that depth-based rendering works for real-life data. To elaborate further, the gradSim approach requires a photorealistic differentiable render that can reproduce images as captured by a real camera. The DIB-R renderer used in this work will compare poorly with real images captured from a vision system. Thus, all gradSim examples only use simulated videos, and the differentiable render relies on flat-shaded renderings of the animations. The work sheds no light on how to handle real-world bias and imperfections. In contrast, our work takes data directly from the real world as raw LIDAR images without any control of lighting conditions. Our differentiable rendering approach is specially tailored to work with depth rendering.

3. Method

We consider a soft body simulation of time t as a *differentiable* function, $\mathcal{S}(t; \mathbf{p})$, which depends on the material parameters $\mathbf{p}$ and returns the simulation state, $\mathbf{s}_t$. Similarly, we define a depth-renderer as a *differentiable* function, $\mathcal{R}(\mathbf{s}_t)$, that projects scene geometry to an image, I_t. Given a set of n target images, $\mathbf{I}^* = \{\mathrm{I}^*_{t_1}, \ldots, \mathrm{I}^*_{t_n}\}$, captured at times, $\mathbf{t} = \{t_1, ..t_n\}$, we leverage the differentiability of our pipeline to compute optimal material parameters, $\mathbf{p}^*$, such that

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \frac{1}{n} \sum_{t \in \mathbf{t}} \|\mathcal{R}(\mathcal{S}(t; \mathbf{p})) - \mathrm{I}^*_t\|_2. \tag{1}$$

In other words, we choose parameters maximizing visual likeliness between a set of real images $\mathbf{I}^*$ and rendered simulation states. The parameter-wise derivatives w.r.t. each target image is, therefore, given by

$$\frac{\partial \|\mathrm{I}_t - \mathrm{I}^*_t\|_2}{\partial \mathbf{p}} = \frac{\partial \mathbf{s}_t}{\partial \mathbf{p}} \cdot \frac{\partial \mathbf{I}_t}{\partial \mathbf{S}_t} \cdot \frac{\partial \|\mathrm{I}_t - \mathrm{I}^*_t\|_2}{\partial \mathbf{I}_t}, \tag{2}$$

which we can use to perform gradient-based optimization of material parameters. An overview of our method can be seen in Figure 2.

3.1. Differentiable rendering

Rendering is not inherently differentiable, as both ray-tracers and rasterization-based graphics pipelines need to deal with discontinuities in the scene. Rendering depth simplifies the problem of differentiability, but problems like occlusion events can still occur. In this work, we use the differentiable ray-tracer presented by Li *et al.* [LADL18], which solves these problems by computing derivatives using a Monte Carlo approach to estimate the pixel integral and the Dirac delta terms at occlusion boundaries. Here, we will briefly summarize the primary idea from Li *et al.*, but we refer the reader to the original paper for a more rigorous description.

The general 2D pixel filter integral describes the pixel colour

$$\mathrm{I}_c = \int \int f(x, y) L(x, y) \mathrm{d}x\mathrm{d}y \tag{3}$$

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

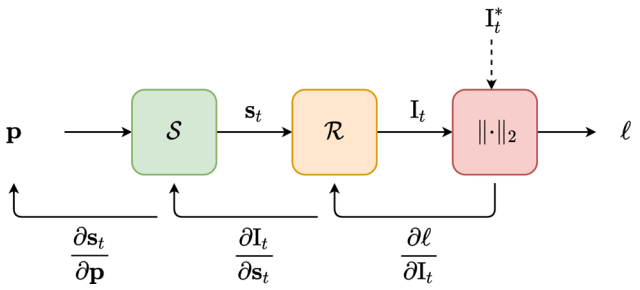


Figure 2: *The data flow of our differentiable pipeline. The simulation function $\mathcal{S}$ produces a state $\mathbf{s}_t$ using material parameters $\mathbf{p}$. The state, $\mathbf{s}_t$, is then projected to an image, I_t, which we compare against a target image, I_t^*, resulting in a scalar loss, ℓ. The chain rule can then be applied to compute the derivative of ℓ w.r.t. material parameters $\mathbf{p}$.*

for pixel filter f and radiance L, but as we will only be rendering depth, we can ignore the radiance term. We are interested in computing pixel-wise derivatives with respect to the simulation state, $\mathbf{s}_t$, so we re-parameterize the pixel filter as $f(x, y; \mathbf{s}_t)$, yielding the state-dependent integral,

$$\mathrm{I}_c = \int\int f(x, y; \mathbf{s}_t)\mathrm{d}x\mathrm{d}y. \tag{4}$$

In this work, we will only be rendering the surface of tetrahedral meshes, which means discontinuities will occur only at triangle edges. Each triangle edge can be seen as splitting the space into two regions: inside and outside of the triangle. Referring to the inside as f^+ and the outside as f^-, the space described by the ith edge can be modelled as:

$$\theta(\alpha_i(x, y; \mathbf{s}))f_i^+(x, y; \mathbf{s}) + \theta(-\alpha_i(x, y; \mathbf{s}))f_i^-(x, y; \mathbf{s}), \tag{5}$$

where θ is the Heaviside step function and α_i is the edge equation of the ith edge. The integral from Equation (4) can therefore be transformed into a sum of Heaviside step functions as

$$\int\int f(x, y; \mathbf{s}_t)\mathrm{d}x\mathrm{d}y = \sum_i \int\int \theta(\alpha_i(x, y; \mathbf{s}_t))f_i(x, y; \mathbf{s}_t)\mathrm{d}x\mathrm{d}y. \tag{6}$$

The derivative of this integral is written as a sum of two terms:

$$\nabla \sum_i \int\int \theta(\alpha_i(x, y; \mathbf{s}_t))f_i(x, y; \mathbf{s}_t)\mathrm{d}x\mathrm{d}y = \tag{7}$$

$$\sum_i \int\int \delta(\alpha_i(x, y; \mathbf{s}_t))\nabla\theta(\alpha_i(x, y; \mathbf{s}_t))f_i(x, y; \mathbf{s}_t)\mathrm{d}x\mathrm{d}y + \tag{7}$$

$$\sum_i \int\int \nabla f_i(x, y; \mathbf{s}_t)\theta(\alpha_i(x, y; \mathbf{s}_t))\mathrm{d}x\mathrm{d}y, \tag{7}$$

where δ is the Dirac delta function. Here, $\nabla f_i(x, y; \mathbf{s}_t)$ can be found by automatic differentiation, but to estimate the gradient of the first integral of Equation (7), we need to eliminate the discontinuous Dirac delta function. The Dirac delta function is 0 everywhere except at the edge dividing the space in half; that is, where $\alpha_i(x, y, \mathbf{s}_t) = 0$. We can, therefore, perform a variable substitution to re-write the integral as

$$\sum_i \int\int \delta(\alpha_i(x, y; \mathbf{s}_t))\nabla\theta(\alpha_i(x, y; \mathbf{s}_t))f_i(x, y; \mathbf{s}_t)\mathrm{d}x\mathrm{d}y = \tag{8}$$

$$\sum_i \int_{\alpha_i(x,y;\mathbf{s}_t)=0} \frac{\nabla\alpha_i(x, y; \mathbf{s}_t)}{\|\nabla_{x,y}\alpha_i(x, y; \mathbf{s}_t)\|_2} f_i(x, y; \mathbf{s}_t)d\sigma(x, y), \tag{8}$$

where $\sigma(x, y)$ is the measure of the length on the edge. For an edge with endpoints (a_x, a_y) and (b_x, b_y), the edge equation is given by

$$\alpha(x, y) = (a_y - b_y)x + (b_x - a_x)y + (a_x b_y - b_x a_y), \tag{9}$$

which means the edge equation derivatives are given by

$$\frac{\partial\alpha}{\partial a_x} = b_y - y, \frac{\partial\alpha}{\partial a_y} = x - b_x, \tag{10}$$

$$\frac{\partial\alpha}{\partial b_x} = y - a_y, \frac{\partial\alpha}{\partial b_y} = a_x - x. \tag{10}$$

We can now propagate the derivatives from the projected triangle vertices as

$$\frac{\partial\alpha}{\partial \mathbf{s}_t} = \sum_{k\in\{x,y\}} \frac{\partial\alpha}{\partial a_k}\frac{\partial a_k}{\partial \mathbf{s}_t} + \frac{\partial\alpha}{\partial b_k}\frac{\partial b_k}{\partial \mathbf{s}_t}. \tag{11}$$

Finally, we can combine the Dirac delta estimated gradients with the derivatives from the continuous term in Equation (7) to obtain the pixel-wise derivatives of our rendered image, I.

3.2. Differentiable physics

Our differentiable soft-body physics simulator is based on the work of Murthy *et al.* [MMG*21]. Our choice is not relevant to the differentiable depth concept that we are introducing. Any differentiable simulator can be used in principle, although the differences in time-stepping are important when considering auto-differentiation. The semi-implicit flavour of our choice can deplete memory and call for taping techniques. Full-implicit can do larger time steps and has no taping thanks to the adjoint method [GHZ*20]. It is a matter of practical trade-offs. Our simulator is simpler to implement and fits easily into auto-diff tools but requires taping due to large GPU memory usage. We do note that it is likely that using the adjoint method could improve the scalability of the method with respect to the number of tetrahedral elements used so that the method could be applied to more complex geometries.

We use the neo-Hookean constitutive model from Smith *et al.* [SGK18], which defines the following elastic strain energy density:

$$\Psi = \frac{\mu}{2}(I_C - 3) + \frac{\lambda}{2}(J - \alpha)^2 - \frac{\mu}{2}\log(I_C + 1). \tag{12}$$

Here λ and μ are the Lamé parameters, and α is a constant. The scalar $J = \det(\mathbf{F})$ is the relative volume change, and $I_C = \mathrm{tr}(\mathbf{F}^T\mathbf{F})$ is the first Cauchy invariant of strain. The matrix $\mathbf{F}$ is the deformation gradient, which is a function of the physical state $\mathbf{s}$ of the system. Integrating the energy density from Equation (12) and summing over each tetrahedral element gives the total elastic potential. We allow each element to have a unique Young's modulus, so each contributes one additional variable to the material parameters $\mathbf{p}$. This allows us

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

to optimize for heterogeneous materials. We model damping using a strain-rate dissipation potential [SBO18] of $\Psi_D = \Psi(\dot{\mathbf{F}})$, where $\dot{\mathbf{F}}$ is the time-derivative of the deformation gradient. We control how much damping we want in the system by a scaling parameter β, which we also add to $\mathbf{p}$.

Forces are derived from the energy gradient analytically and integrated using a semi-implicit Euler. To compute gradients of the state with respect to material parameters, we use a source-code transformation approach that generates parallel GPU kernels for reverse mode auto-differentiation [HAL*20]. The use of semi-implicit integration places some time-step size restrictions on our method that can lead to large memory requirements when simulating long time spans. To alleviate this, we use checkpointing [GW08], which replays some computations from snapshots of the state, trading additional computation for reduced memory use.

Conceptually, our physics simulator provides a forward mapping from some initial state to the final state $\mathbf{s}$ at a time t in the future. In addition, it provides a reverse mapping for the gradients by evaluating the following:

$$\frac{\partial \ell}{\partial \mathbf{s}} \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{p}} = \frac{\partial \ell}{\partial \mathbf{p}}. \tag{13}$$

Combining this with our differentiable renderer, we can use the chain rule to compute gradients of our entire pipeline with respect to material parameters as follows:

$$\frac{\partial \ell}{\partial \mathrm{I}} \cdot \frac{\partial \mathrm{I}}{\partial \mathbf{s}} \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{p}} = \frac{\partial \ell}{\partial \mathbf{p}}, \tag{14}$$

where ℓ is a scalar loss. This allows us to use gradient-based optimization to minimize ℓ and fit simulated material parameters to real-world data.

4. Experiments and Results

We validate our method on a set of real silicone test objects. For all objects, we design and 3D-print multi-part moulds. Small dowels are added to each mould part and clamps are used to decrease the likelihood of leakage. For silicone, we use Ecoflex-50 and MoldStar-15, which we colour red and blue to make the materials more distinguishable.

We place our test objects in a cube test environment similar to the set-up of Holsten *et al.* [HENDE19] and force them into desired configurations using 3D-printed clamps. We capture data using a single Intel L515 LIDAR scanner. See Figure 3 for an example of how the soft objects were placed in the cube. The clamps are mapped directly to Dirichlet boundary conditions, similar to Hahn *et al.* [HBBC19]. To match the rendered scene to the physical set-up, we estimate the camera pose using ArUco [GJMSMCMJ14] markers and construct our projection matrix using the intrinsic parameters of the L515 camera. We place the ArUco marker on the robot clamp, such that we can easily place the soft object in the virtual scene.

For all experiments, we use the Adam optimizer with decaying momentum [CWLK22] using parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate, η, scaled such that the sensitivity of the input parameters and the output loss is similar. In other words, we select η to ensure that the updated parameters produce an observable visual change in the rendered image. For all experiments, the goal of the optimization is to minimize the L2 norm of the difference between target and rendered images. As we compare the rendered images to the raw depth data, the optimal material parameters might not result in a loss of 0, but instead a constant value related to the amount of non-zero background pixels in the target image. We implement the loss function and optimizer using PyTorch [PGM*19] to benefit from its automatic differentiation framework. For each object, we show the mean convergence rate of 10 repeated experiments with initial Young's modulus chosen randomly within ±35–45% of the minimizer. For simplicity, we assume that our materials have a Poisson's ratio of 0.49 for all of our experiments. In principle, one can optimize for the Poisson ratio too. It is worth noting that calibration is often a pre-process and not a run-time system. Hence, we do not worry about absolute timings but nevertheless, report on iteration counts and convergence behaviours. Table 1 provides an overview of our experiments and results.

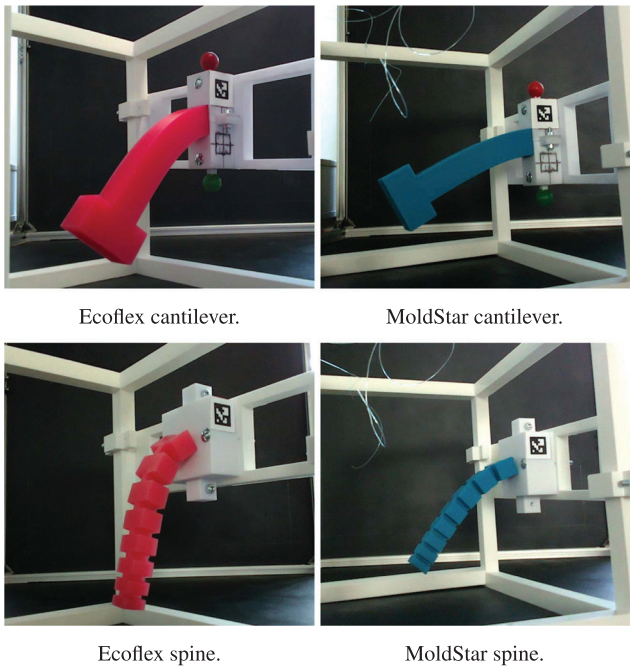


Figure 3: *The physical set-up for the simple homogeneous soft body experiments. Camera angles were chosen arbitrarily, with the only requirement that the ArUco markers had to be visible.*

Passive deformation. We start by demonstrating that our differential depth pipeline applies to the simple homogeneous objects from Figure 3 in passive set-ups subject only to gravity. For this experiment, the material model is parameterized by a single Young's modulus, which is used to compute the Lamé parameters of our neo-Hookean model. We expect to see convincing convergence towards solutions that match real-depth images. Our cases include variation in camera angles, lighting conditions, shape diversity and materials. The convergence plot of the loss functions can be found in Figure 4. As expected, the optimizer converges in a few iterations for all of the randomly initialized runs. Figure 5 shows a visual comparison between sample starting iterates and minimizers for the MoldStar-

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

Table 1: *Summary of experimental setup and results. Row numbers 1–10 and 11–12 correspond to passive deformation and viscosity estimation experiments, respectively. All experiments are repeated 10 times with random initial parameters. Gravity is applied in all of the experiments. We use silicone in all experiments and keep the Poisson's ratios fixed at 0.49.*

#	Corresponding Figures	Silicone Material	Shape	Deformation	Optimized Parameters	Learning Rates	# Iterations	Mean Optimal Parameters
1	(3 and 4) top left	Ecoflex	Cantilever	Hanging	Global Young's modulus	1e6	50	65 kPa
2	(3 and 4) bottom left		Spine			1e3	50	44 kPa
3	12		XYZ RGB Dragon			5e3	25	95 kPa
4	(3 and 4) top right, 5 top row	MoldStar	Cantilever			1e6	30	149 kPa
5	(3 and 4) bottom right, 5 bottom row		Spine			1e3	5	110 kPa
6	(6 and 7 top row)	Ecoflex+MoldStar	Cantilever	Hanging	Global Young's modulus for each material	1e3	42	(77, 170) kPa
7	8 top row				Tetrahedron-wise Young's moduli	1e4	7	—
8	(6 and 7) bottom row			Twisting	Global Young's modulus for each material	1e3	23	(119, 217) kPa
9	8 bottom row				Tetrahedron-wise Young's modulus	1e7	26	—
10	9			Hanging+Twisting	Tetrahedron-wise Young's modulus	1e6	50	—
11	11 right	MoldStar	Coarse cantilever	Oscillating	Young's modulus, damping factor, mass density	(1e3, 1e-1, 1)	8	223 kPa, 11.1, 1050 kg/m³
12	10 11 left		Fine cantilever			(1e3, 1e-1, 1)	10	277 kPa, 8.8, 1057 kg/m³

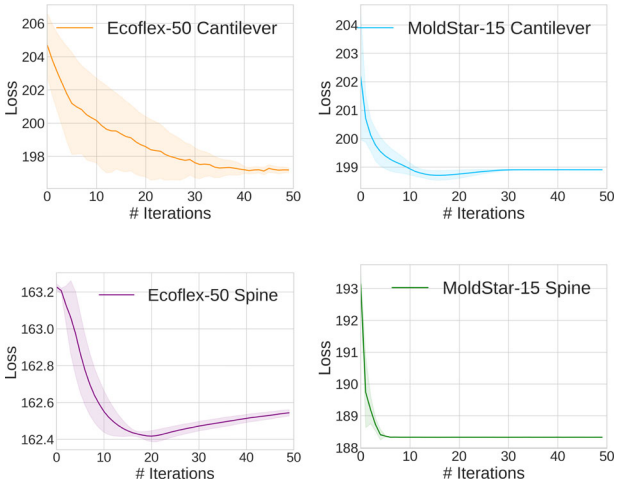


Figure 4: *Convergence for the cantilever and spine fabricated using Ecoflex-50 and MoldStar-15. For all four soft objects, we used 10 random initial parameters. The mean of the loss function is shown in bold and its one standard deviation as shaded regions.*

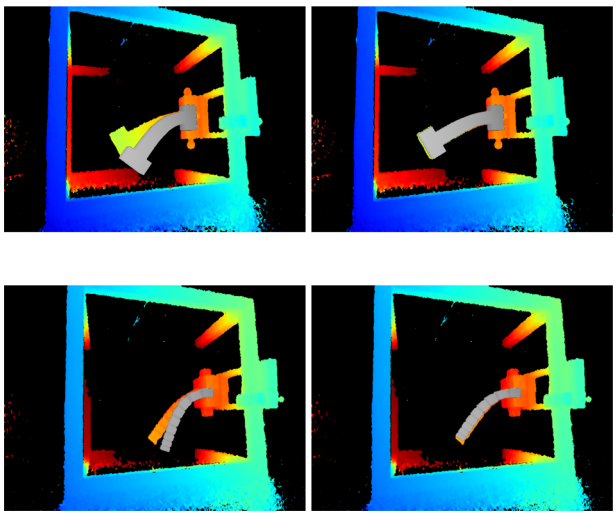


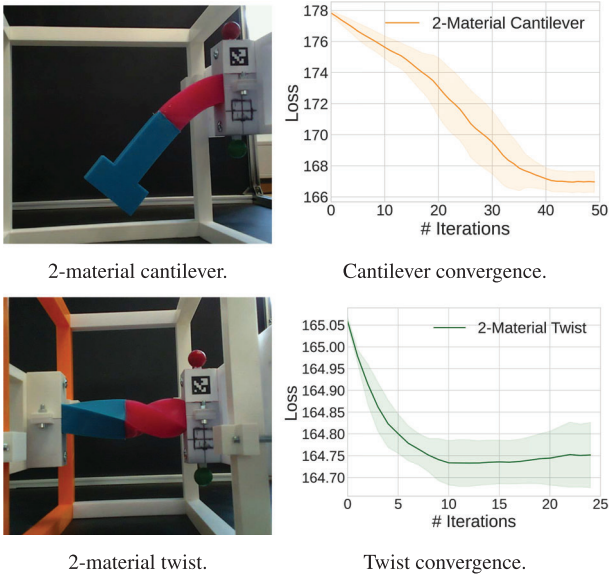
Figure 5: *A visual comparison between starting iterates (left) and minimizers (right) for the MoldStar-15 cantilever and spine. In all images, the simulated cantilever is rendered in grey.*

15 cantilever and spine experiments. Here, we show the depth image captured by the L515 camera with the rendered starting iterates and minimizers imposed on top in grey. The digital cantilever consists of 1750 elements, while the spine model has 1327 tetrahedra. For the Ecoflex-50 experiments, we found the mean Young's modulus of the minimizers to be 65 and 44 kPa for the cantilever and spine, respectively. The mean Young's modulus of the minimizers for the MoldStar-15 experiments was given by 149 kPa for cantilever and 110 kPa for spine model.

Heterogeneous materials. Our method easily extends to heterogeneous materials and can also be applied to larger deformations. We fabricate a two-part cantilever and subject it to the same set-up as its homogeneous counterparts. We then force it into a more stressful deformation by twisting the previously free-hanging part 180º. The two set-ups can be found in Figure 6. For both experiments, we parameterize the elastic material model by two Young's modulus values instead of one, such that each half of the cantilever has its own parameter. The outcome of running 10 experiments from different initial values for the hanging and twisted cantilever can be observed in Figure 6. For all 10 experiments, the two material parameters were initially assigned the same value. In Figure 7, visual inspections of sample starting iterates and minimizers can be found for each experiment. The same digital model of the cantilever was used for both experiments and consisted of 3414 elements. The mean minimizer of the hanging cantilever was found to be 170 and 77 kPa for the MoldStar-15 and Ecoflex-50 parts, respectively. For the twisted cantilever, the mean minimizer was given by 217 and 119 kPa.

In many fabricated objects, where materials have been mixed unevenly or contain internal structures, manually designing correct material distributions is infeasible. Our method handles this problem by optimizing directly for tetrahedron-wise material parameters. We repeat the 2-material cantilever experiments, but this time optimizing for the tetrahedron-wise parameters. For both experiments, we use a digital model of the cantilever consisting of 3414 tetrahedra. In Figure 8, we show an example of a distribution of Young's modulus for minimizers of both experiments as well

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.



2-material cantilever. Cantilever convergence.

2-material twist. Twist convergence.

Figure 6: *Physical set-up for the 2-material composite cantilever experiments and their respective material parameter convergence plots. Notice that plots show 10 experiments with random initial parameters; the bold lines show the mean of the loss function and shaded regions show one standard deviation.*

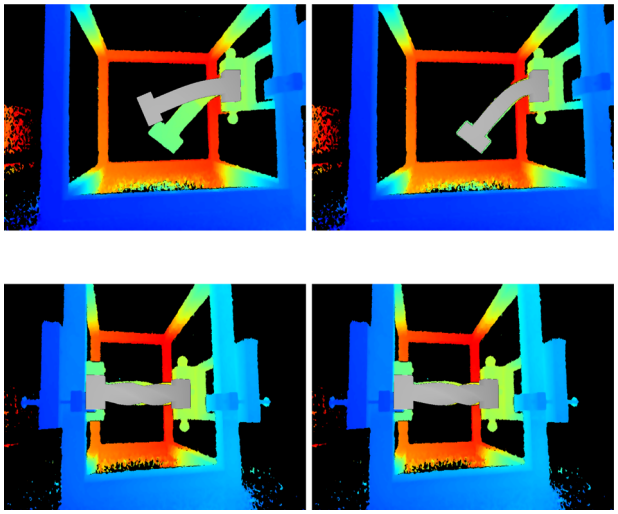


Figure 7: *A visual inspection of the 2-material experiments. In the top row, we visualize a starting iterate and minimizer for the hanging cantilever, and in the bottom row, we visualize a starting iterate and minimizer for the twisting experiment. For both experiments, the simulated cantilever is rendered in grey.*

as the convergence plots. The aim of our experiment is not to get smooth real-world parameter fields. The point of our twist experiment is to demonstrate that we can handle heterogeneous solids. A regularizer could be used as a model to achieve smoother solutions, or the tet-wise values could be interpolated from some coarse sampling to add more model semantics to the solution. Such extensions

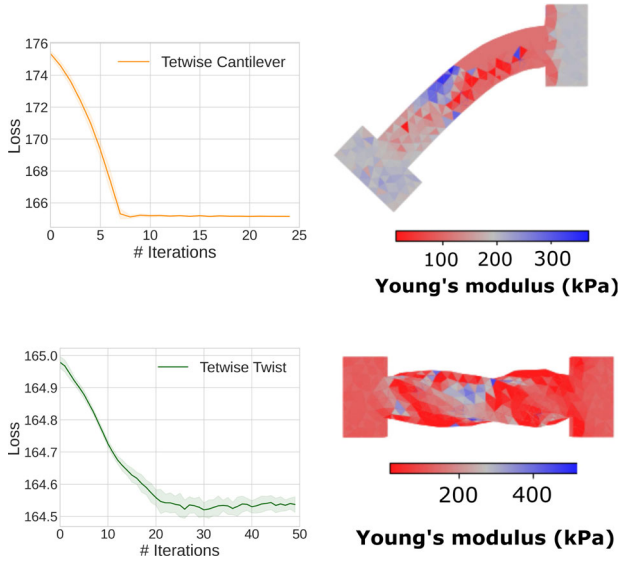


Figure 8: *The result of optimizing the material parameters of each tetrahedron for the hanging and twisted cantilever. The resulting tetrahedron-wise distribution of Young's modulus for randomly picked minimizers has been visualized next to the convergence plots. Observe the very small standard deviation (shaded regions) of our 10 experiments.*

can be added as an extra penalty term to our loss function, or as an extra derivative term to our pipeline. The fact that we get a more speckled distribution underlines that one can not expect to throw real-world values into a fast simulator to match the real world.

From Figures 6 and 8, we note that optimizing for tetrahedron-wise parameters results in a similar minimum function value as the low-parameter counterparts. For both cases, we also observe a less noisy optimization for the tetrahedron-wise experiments, which indicates that the optimizer benefits from having more tunable parameters.

Generalizing to different deformation modes and boundary conditions. In our experiments, so far we have optimized for only a single deformation mode. We obtain different results for the same models and materials depending on the changes in the setup. This is to be expected because the observed individual deformations make us fit the simulator to different principal deformation modes of the physical models. Changing the experimental setup or boundary conditions will lead to different results as we are fitting to different deformation modes. This insight is related to how subspace modelling approaches work. One seeks a subset of individual modes from a general model, each mode has its own 'stiffness' given by a singular value or similar. When calibrating a simulator, we have the reverse scenario. We should expose the simulator to data examples—aka deformation modes—that span the general behaviour one wishes the simulator to cover with high fidelity. This feat is shared by all parameter estimation methods regardless of them being marker-based or not.

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

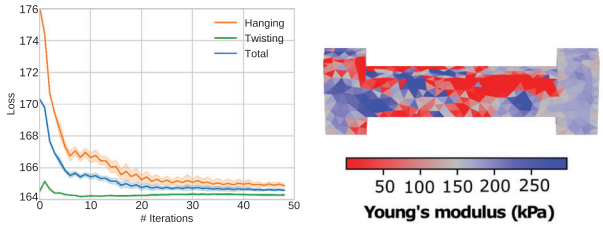


Figure 9: *Optimizing for both bending and twisting deformations simultaneously. On the left, the convergence plot of each loss function and the combined one is shown. On the right, we have the final distribution of tet-wise Young's modulus values. As expected, we obtain a different solution from that of Figure 8, which is optimized on single modes.*

Our approach does allow for an easy combination of different modes by simply adding loss functions from the individual models to obtain a combined loss function. We demonstrate this by combining the bending and twisting cantilever beam examples into one where we optimize for the shared tet-wise Young's modulus to make the simulator match both modes at the same time. Note that both boundary conditions and deformations are different in the two examples. Figure 9 shows the convergence plots as well as the final distribution of the Young modulus. The undeformed coordinates are used for the visualization of Young's modulus solution. As expected, we obtain a different solution from those of the single-mode optimizations.

Viscosity estimation. We wish to show that our method can also be applied to calibrate the viscous behaviour of soft objects. To handle the calibration of a dynamic motion, we need data on the soft object in motion. During optimization, we then continuously render the state of our simulation, corresponding to the captured frames of the motion. We use a similar set-up to the one we used to capture static deformations, but add a small ledge that can be removed by pulling a trigger. This allows us to match the beginning of our simulation to the frame where the ledge was removed.

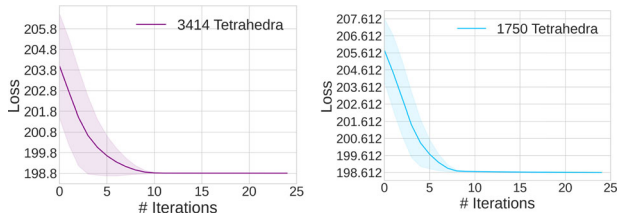


Figure 10: *The convergence of the viscosity calibration experiment for the fine (left) and coarse (right) cantilever models. Experiments were repeated 10 times with initial random parameters. The mean of the loss is in bold, and shaded regions show one standard deviation.*

We calibrate for correct viscous behaviour by allowing the optimizer to change Young's modulus of the material, the damping parameter described in Section 3.2 and the density of the material. We estimate these three parameters for the MoldStar-15 cantilever and run the experiment 10 times using randomly initialized parameters. The loss is the mean of the L2-norms of the difference between target images and rendered simulation states. For this experiment, 12 images were captured at 30 Hz, resulting in roughly 0.4 s of motion. We perform this experiment for two differently discretized cantilever models; one consisting of 1750 elements and one consisting of 3414. We show the mean convergence of 10 randomly initialized experimental runs in Figure 10, and perform a visual inspection of a randomly selected starting iterate and minimizer pair in Figure 11 for the coarsest mesh. For the coarse mesh, the mean minimizer had Young's modulus of 223 kPa, a damping factor, β, of 11.1 and a density of 1050 kg/m^3. On the other hand, the mean minimizer of the fine mesh was given by Young's modulus of 277 kPa, a damping factor of 8.8 and a density of 1057 kg/m^3.

Complex shapes. So far, the soft bodies in our experiments have consisted of simple symmetric designs. As our method relies on optimizing for visual likeness between rendered states and captured images, we expect our method to work at least as well for more complex objects. This is an important feature of the differentiable

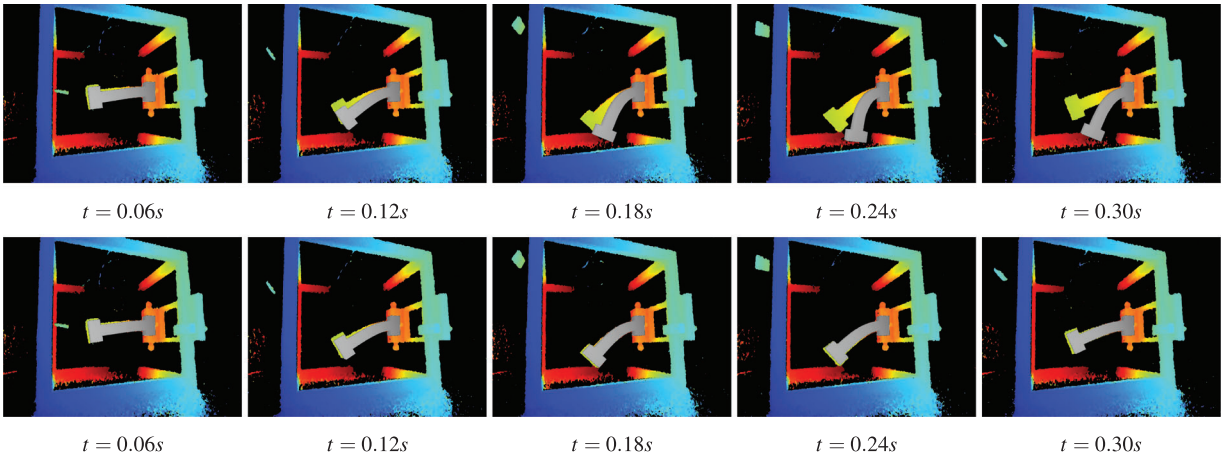


Figure 11: *Visualizations of simulations of a starting iterate (top) and minimizer (bottom) for the dynamic motion experiment.*

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

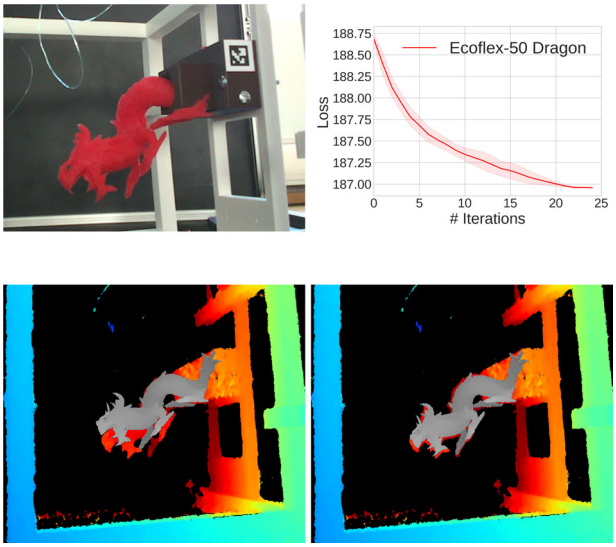


Figure 12: *The XYZ RGB Dragon set-up and the convergence of the material parameter optimization, above an example of a starting iterate and minimizer of the optimization. We show the mean of the loss function in bold and its one standard deviation as shaded regions over 10 random initial parameters.*

pipeline, as it will otherwise be unsuited for performing simulator calibration for real fabricated objects. We explore this robustness towards complex shapes with potential self-occlusions by calibrating a simulation of an Ecoflex-50 version of the XYZ RGB Dragon subject to gravity. The set-up and the results can be found in Figure 12. Note that while the loss function only decreases by a small amount, we still observe a noticeable change in the deformation of the dragon. The mean Young's modulus of the minimizer for the XYZ RGB Dragon experiment was 95 kPa.

Numerical coarsening. The differentiable depth approach can be applied to fit coarse meshes suffering from numerical stiffness to the behaviour of high-resolution meshes. We use a rendered image of a high-resolution XYZ RGB Dragon to replace the LIDAR depth image and apply our technique to the coarsened versions of the Dragon. Dirichlet boundary conditions are placed on the Dragon from the hind legs to the tip of the tail, and the target image was generated using Young's modulus of 65 kPa. We anticipate our approach to decrease Young's modulus of the coarser meshes, in order to counteract the increasing numerical stiffness. The computational footprint should increase w.r.t. the number of elements, which will reflect on both the computational time of the simulator and the optimizer. The convergence behaviour should be indifferent to the coarseness of the Dragon as the gradient is largely given by the difference between the rendered target and the iterate. The resulting deformations and minimizers can be found in Figure 13. As seen from the listed results, there is a good agreement between the measurements and our hypotheses.

Effect of different material models. The choice of the material model can have a huge influence on the material parameter values. In general, material parameters are not universal constants but rather depend on the material model, i.e. the constitutive equation used. Hence, one cannot simply swap a Young's modulus value from a linear material model into a non-linear one and expect to get the same material behaviour.

Our framework optimizes the parameters of any material model to make the simulated deformations as close as possible to the observed deformations from the depth images. We note that the silicone we use has a material behaviour that corresponds well with the stable neo-Hookean material model. To demonstrate the influence of the choice of the material model, we have compared results with Mooney–Rivlin and linear elasticity models. Figure 14 shows that the obtained deformations are as close to the observations as the models allow but will have different material parameter values depending on the model choice. The optimized Young's modulus for the neo-Hookean, Mooney–Rivlin and linear elastic models are 57, 95 and 150 kPa, respectively. As expected, we obtained quite different values.

The deformations obtained from the simulation can be different if one uses different models. The choice of model essentially dictates the feasible sub-space of motions that can be achieved. As a result, a poor choice of model may cause the optimization not to reproduce the observed deformation very well, as seen in the middle and bottom row of Figure 14 to varying degrees.

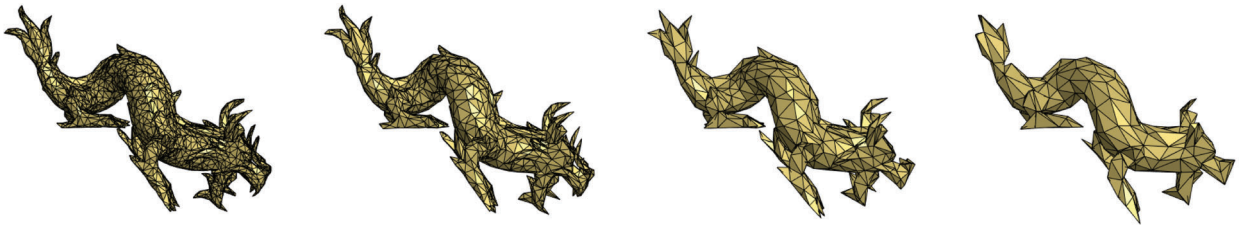
Beyond elasticity. Lastly, we examine how our approach extends to optimize for parameters other than elasticity. We focus on cable control for soft robotic fingers, and therefore, we have added cables to a soft finger suspension. We expect our method to visually achieve similar configurations as the real finger by optimizing for cable rest length. Figure 15 shows the target configuration, the minimizer and the convergence of the optimization for three increasingly difficult cable pulls. We observe a good match between simulated and real robot fingers. Further, we notice the optimization problem becomes more difficult for smaller cable lengths. This is expected, due to the larger deformations, but our method still handles the challenge gracefully.

5. Limitations and Discussion

In our experiments, it was sufficient to use a single off-the-shelf LIDAR camera. LIDAR is active light, and therefore, the reflectance of dissipative media can create noise near the silhouette of objects. Further, the inexpensive equipment has limitations in focal length which are important for getting sufficiently high-resolution data. Our setup was robust towards these potential limitations in hardware and worked well under different light conditions. Alternatively, one may apply object coating if reflectance noise is a huge concern or apply another type of depth sensor technology such as stereo vision with specialized lens systems. The benefits of our choice are its inexpensiveness and straightforwardness.

All our experiments take full advantage of standard optimization software on simple L2-norms of depth images and differential depth

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.



# Elements	7913	2415	1023
Single simulation	145.1 seconds	74.71 seconds	19.45 seconds
Optimization runtime	65.3 minutes	31.13 minutes	9.4 minutes
# Function evaluations	27	25	29
Young's modulus	46.19 kPa	42.84 kPa	37.74 kPa

Figure 13: *The XYZ RGB Dragon deformed under gravity for a set of meshes with varying coarseness. The Dragon is kept in place by Dirichlet conditions placed on the left part of the abdomen, feet and tail. In the left-most column, a mesh of 17,827 elements can be found, which was used to render a target depth image. The target dynamic simulation has Young's modulus of 65 kPa, and was run until it reached static equilibrium after 293.18 s. The Young's modulus of the material was optimized for each of the coarser meshes in order to minimize the L1-norm between the target image and the final rendered simulation state. All mesh resolutions arrived at a minimizer within a small number of function evaluations and as expected, arrived at lower optimal material parameters due to numerical stiffness.*

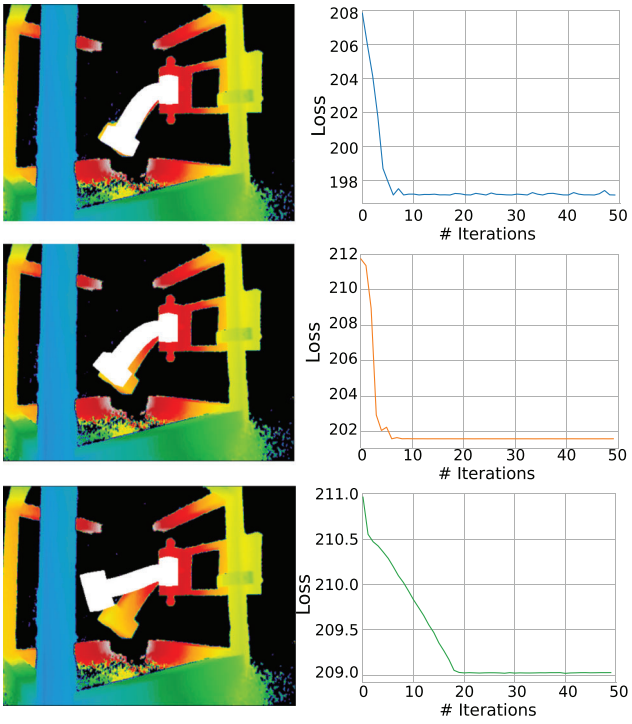


Figure 14: *Comparison of various constitutive models for the hanging Ecoflex cantilever experiment. Top row: stable neo-Hookean model; middle row: Mooney-Rivlin model; bottom row: linear elasticity model. The depth images shown in colour and rendered object is overlaid in white. Clearly, one must choose a model which is well-matched to the type of material being simulated. Having said that our framework brings the rendered image as close to the depth image as the chosen model allows it to. In doing so, one would obtain different values for material parameters in each model.*

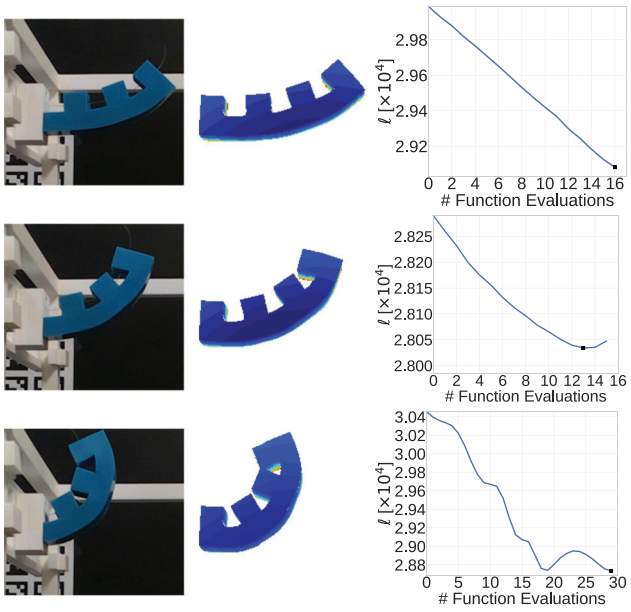


Figure 15: *Convergence of the soft robotic finger actuated by a cable. From top to bottom the cable was pulled 18, 31 and 46 mm. Minima are marked by black squares. The soft finger was fabricated using MoldStar-15. The visual accuracy of our method is verified by the depth rendered versions of the finger, actuated according to the minimizer of the objective function.*

render differences. This makes markers and point correspondences unneeded and thus does not require large data sets for visual appearance models. In this work, the calibration of the depth sensor and renderer was handled using ArUco markers, but any other solutions to camera pose estimation can easily be integrated into our pipeline.

Our cost function is fully differentiable. As we used PyTorch, Adam was a straightforward choice as an optimizer, but other line-

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

search strategies could also be applied. Unlike the depth images, the rendered images contain only the object and not the background, meaning the loss cannot be minimized to zero. The larger this background noise, the more difficult it would be to reduce the cost function. Further, the pixel resolution will affect results too, as we optimize until the pixel difference is small enough.

The depth information provides a lot of extra information for handling complex geometries as evidenced by our dragon and twisting cantilever examples. This, combined with the use of a physical simulator, addresses occlusion challenges to a certain extent, which are known limitations in point cloud or image/video-based approaches. One classical remedy to occlusion is to add more cameras. We remark that our approach is not technically limited to a single depth camera and can easily be extended to multiple cameras in future work by rendering from multiple angles.

We exploit that all our silicone objects are fabricated and that the reference shapes for all our experiments are known. An interesting result from our heterogeneous experiments is that the same deformation can be reached by very different material distributions. Especially for the twisting cantilever, we observed an unexpected material distribution in the tet-wise optimization experiment. We also note that the material parameters depended heavily on the discretization of the soft objects. These results emphasize the need for a fast way to re-calibrate material parameters, as we cannot rely on values obtained from traditional parameter estimation set-ups to translate to correct simulated behaviour.

In our dynamic calibration experiment, we had to restrict the time span of the motion due to GPU memory constraints. In future work, we would like to solve this issue, such that we can apply our method to longer motions, for instance by using the adjoint method. The time span of the captured data has an obvious effect on the number of iterations needed to calibrate parameters. However, the convergence rate of the optimizer should be independent of the number of target frames, as long as the build-up of errors from the semi-implicit solver remains sufficiently small. If calibration of mass density and damping is less relevant, then static simulations could be considered to improve performance.

6. Conclusion

We have demonstrated a differentiable depth calibration method that is inexpensive and easy to set up and use. It is built on the simple principles of back-propagation to evaluate gradients. The method relies on image-based loss functions, avoiding the need for textures, markers or point correspondences. This novel differentiable depth estimation method allows for a vast class of gradient-based optimizers. Our example cases range from passive homogeneous set-ups to more complicated shapes and heterogeneous objects. We robustly estimate elasticity, density and viscous behaviour. We investigated the impact of numerical coarsening on the estimated values. Finally, we used our framework to optimize a cable length of a soft robotic finger. The resulting workflow allows for a very agile and robust approach to simulator calibration for fabricated objects.

Acknowledgement

This project has received funding from Independent Research Fund Denmark (DFF) under agreement No. 9131-00085B.

References

[BBO*09] Bickel B., Bächer M., Otaduy M. A., Matusik W., Pfister H., Gross M.: Capture and modeling of non-linear heterogeneous soft tissue. *ACM Transactions on Graphics (TOG) 28*, 3 (2009), 1–9.

[BBPC19] Bern J. M., Banzet P., Poranne R., Coros S.: Trajectory optimization for cable-driven soft robot locomotion. In *Robotics: Science and Systems XV* (Freiburg im Breisgau, Germany, 2019), A. Bicchi, H. Kress-Gazit and S. Hutchinson (Eds.), University of Freiburg.

[BCC17] Bern J. M., Chang K.-H., Coros S.: Interactive design of animated plushies. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–11.

[BJ05] Barbič J., James D. L.: Real-time subspace integration for st. Venant–Kirchhoff deformable models. *ACM Transactions on Graphics 24*, 3 (July 2005), 982–990.

[BKS*12] Bickel B., Kaufmann P., Skouras M., Thomaszewski B., Bradley D., Beeler T., Jackson P., Marschner S., Matusik W., Gross M.: Physical face cloning. *ACM Transactions on Graphics 31*, 4 (July 2012), 1–10.

[BSB*20] Bern J. M., Schnider Y., Banzet P., Kumar N., Coros S.: Soft robot control with a learned differentiable model. In *Proceedings of the 2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)* (2020), pp. 417–423.

[BT07] Becker M., Teschner M.: Robust and efficient estimation of elasticity parameters using the linear finite element method. In *Proceedings of the Simulation und Visualisierung 2007 (SimVis 2007)* (Magdeburg, 2007), T. Schulze, B. Preim and H. Schumann (Eds.), SCS Publishing House e.V., pp. 15–28.

[BTH*03] Bhat K. S., Twigg C. D., Hodgins J. K., Khosla P. K., Popović Z., Seitz S. M.: Estimating cloth simulation parameters from video. In *SCA'03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2003), Eurographics Association, pp. 37–51.

[CLG*19] Chen W., Ling H., Gao J., Smith E., Lehtinen J., Jacobson A., Fidler S.: Learning to predict 3D objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems* (2019), pp. 9609–9619.

[CLMK17] Chen D., Levin D. I., Matusik W., Kaufman D. M.: Dynamics-aware numerical coarsening for fabrication design. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–15.

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

[CLSM15] Chen D., Levin D. I. W., Sueda S., Matusik W.: Data-driven finite elements for geometry and material design. *ACM Transactions on Graphics 34*, 4 (July 2015), 1–10.

[CWLK22] Chen J., Wolfe C., Li Z., Kyrillidis A.: Demon: Momentum decay for improved neural network training. In 2022 IEEE International Conference on Acoustics, Speech, and Signal Processsing (ICASSP) (2022), pp. 3958–3962 (2022).

[dABPSA*18] de Avila Belbute-Peres F., Smith K., Allen K., Tenenbaum J., Kolter J. Z.: End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems* (2018), S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett (Eds.), vol. *31*, Curran Associates, Inc., pp. 7178–7189.

[DHDw19] Degrave J., Hermans M., Dambre J., wyffels F.: A differentiable physics engine for deep learning in robotics. *Frontiers in Neurorobotics 13* (2019), 6.

[GHZ*20] Geilinger M., Hahn D., Zehnder J., Bächer M., Thomaszewski B., Coros S.: Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics 39*, 6 (Nov. 2020), 1–15.

[GJMSMCMJ14] Garrido-Jurado S., Muñoz-Salinas R., Madrid-Cuevas F. J., Marín-Jiménez M. J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition 47*, 6 (2014), 2280–2292.

[GW08] Griewank A., Walther A.: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2008.

[HAL*20] Hu Y., Anderson L., Li T.-M., Sun Q., Carr N., Ragan-Kelley J., Durand F.: DifTaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations* (2020). https://openreview.net/forum?id=B1eB5xSFvr.

[HBBC19] Hahn D., Banzet P., Bern J. M., Coros S.: Real2sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–13.

[HENDE19] Holsten F., Engell-Nørregård M. P., Darkner S., Erleben K.: Data driven inverse kinematics of soft robots using local models. In *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 6251–6257.

[HLS*19] Hu Y., Liu J., Spielberg A., Tenenbaum J. B., Freeman W. T., Wu J., Rus D., Matusik W.: Chainqueen: A real-time differentiable physical simulator for soft robotics. In *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)* (2019), pp. 6265–6271.

[HMZS20] Heiden E., Millard D., Zhang H., Sukhatme G. S.: Interactive differentiable simulation. http://arxiv.org/abs/1905.10706 (2020). Accessed: 15 January 2022.

[ITF04] Irving G., Teran J., Fedkiw R.: Invertible finite elements for robust simulation of large deformation. In *SCA'04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2004), Eurographics Association, pp. 131–140.

[KAMS20] Kandukuri R. K., Achterhold J., Möller M., Stückler J.: Learning to identify physical parameters from video using differentiable physics. http://arxiv.org/abs/2009.08292 (2020). Accessed: 15 January 2022.

[LADL18] Li T.-M., Aittala M., Durand F., Lehtinen J.: Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–11.

[LB14] Loper M. M., Black M. J.: OpenDR: An approximate differentiable renderer. In *Proceedings of the European Conference on Computer Vision* (2014), Springer, pp. 154–169.

[LB15] Li Y., Barbič J.: Stable anisotropic materials. *IEEE Transactions on Visualization and Computer Graphics 21*, 10 (2015), 1129–1137.

[LL20] Liang J., Lin M. C.: Differentiable physics simulation. In *Proceedings of the ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations* (2020). https://openreview.net/forum?id=p-SG2KFY2.

[LLK19] Liang J., Lin M., Koltun V.: Differentiable cloth simulation for inverse problems. In *Proceedings of the Advances in Neural Information Processing Systems* (2019), H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett R. (Eds.), vol. *32*, Curran Associates, Inc., pp. 772–781.

[LLKL*20] Le Lidec Q., Kalevatykh I., Laptev I., Schmid C., Carpentier J.: Differentiable Simulation for Physical System Identification. Working paper or preprint, Nov. 2020. https://hal.archives-ouvertes.fr/hal-03025616.

[MBT*12] Miguel E., Bradley D., Thomaszewski B., Bickel B., Matusik W., Otaduy M. A., Marschner S.: Data-driven estimation of cloth simulation models. *Computer Graphics Forum (Proc. of Eurographics) 31*, 2 (May 2012), 519–528.

[MEM*20] Macklin M., Erleben K., Müller M., Chentanez N., Jeschke S., Kim T. Y.: Primal/dual descent methods for dynamics. In *SCA'20: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2020), Eurographics Association.

[MMG*21] Murthy J. K., Macklin M., Golemo F., Voleti V., Petrini L., Weiss M., Considine B., Parent-Lévesque J., Xie K., Erleben K., Paull L., Shkurti F., Nowrouzezahrai D., Fidler S.: gradSim: Differentiable simulation for system identification and visuomotor control. In *Proceedings of the International Conference on Learning Representations* (2021). https://openreview.net/forum?id=c_E8kFWfhp0.

[MTB*13] Miguel E., Tamstorf R., Bradley D., Schvartzman S. C., Thomaszewski B., Bickel B., Matusik W., Marschner

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

S., Otaduy M. A.: Modeling and estimation of internal friction in cloth. *ACM Transactions on Graphics 32*, 6 (Nov. 2013), 1–10.

[MTGG11] Martin S., Thomaszewski B., Grinspun E., Gross M.: Example-based elastic materials. *ACM Transactions on Graphics 30*, 4 (July 2011), 1–8.

[PGM*19] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Kopf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang L., Bai J., Chintala S.: PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the Advances in Neural Information Processing Systems 32* (2019), H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett (Eds.), Curran Associates, Inc., pp. 8024–8035.

[PKLD20] Piovarči M., Kaufman D. M., Levin D. I. W., Didyk P.: Fabrication-in-the-loop co-optimization of surfaces and styli for drawing haptics. *ACM Transactions on Graphics 39*, 4 (July 2020), 116:1–116:16.

[SB20] Song C., Boularias A.: Learning to slide unknown objects with differentiable physics simulations. In *Proceedings of the Robotics: Science and Systems (R:SS)* (Corvallis, Oregon, USA, 2020), Oregon State University.

[SBO18] Sánchez-Banderas R. M., Otaduy M. A.: Strain rate dissipation for elastic deformations. *Computer Graphics Forum (Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation) 37*, 8 (2018), 161–170.

[SF18] Schenck C., Fox D.: Spnets: Differentiable fluid dynamics for deep neural networks. In *Proceedings of The 2nd Conference on Robot Learning* (Oct 2018), A. Billard, A. Dragan, J. Peters and J. Morimoto (Eds.), vol. *87* of *Proceedings of Machine Learning Research*, PMLR, pp. 317–335.

[SGK18] Smith B., Goes F. D., Kim T.: Stable neo-Hookean flesh simulation. *ACM Transactions on Graphics 37*, 2 (Mar. 2018), 1–15.

[SLK*20] Sengupta A., Lagneau R., Krupa A., Marchand E., Marchal M.: Simultaneous tracking and elasticity parameter estimation of deformable objects. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 10038–10044.

[SNW20] Sperl G., Narain R., Wojtan C.: Homogenized yarn-level cloth. *ACM Transactions on Graphics 39*, 4 (July 2020), 48:1–48:15.

[STC*13] Skouras M., Thomaszewski B., Coros S., Bickel B., Gross M.: Computational design of actuated deformable characters. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–10.

[VCO20] Verschoor M., Casas D., Otaduy M. A.: Tactile rendering based on skin stress optimization. *ACM Transactions on Graphics 39*, 4 (July 2020), 90:1–90:13.

[WAB20] Wang K., Aanjaneya M., Bekris K.: An end-to-end differentiable but explainable physics engine for tensegrity robots: Modeling and control. http://arxiv.org/abs/2011.04929 (2020). Accessed: 15 January 2022.

[WDK*20] Wang B., Deng Y., Kry P., Ascher U., Huang H., Chen B.: Learning elastic constitutive material and damping models. *Computer Graphics Forum 39*, 7 (2020), 81–91.

[WOR11] Wang H., O'Brien J. F., Ramamoorthi R.: Data-driven elastic models for cloth: Modeling and measurement. In *SIGGRAPH'11: ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), Association for Computing Machinery.

[WWY*15] Wang B., Wu L., Yin K., Ascher U. M., Liu L., Huang H.: Deformation capture and modeling of soft objects. *ACM Transactions on Graphics 34*, 4 (2015), 94.

[WZB17] Wang B., Zhao Y., Barbič J.: Botanical materials based on biomechanics. *ACM Transactions on Graphics 36*, 4 (July 2017), 1–13.

[XB17] Xu H., Barbič J.: Example-based damping design. *ACM Transactions on Graphics 36*, 4 (July 2017), 1–14.

[XLCB15] Xu H., Li Y., Chen Y., Barbič J.: Interactive material design using model reduction. *ACM Transactions on Graphics 34*, 2 (Mar. 2015), 1–14.

[XSZB15] Xu H., Sin F., Zhu Y., Barbič J.: Nonlinear material design using principal stretches. *ACM Transactions on Graphics 34*, 4 (July 2015), 1–11.

[YL16] Yang S., Lin M. C.: Bayesian estimation of non-rigid mechanical parameters using temporal sequences of deformation samples. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 4036–4043.

[YLL17] Yang S., Liang J., Lin M. C.: Learning-based cloth material recovery from video. In *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 4393–4403.

[YLYW18] Yan G., Li W., Yang R., Wang H.: Inexact descent methods for elastic parameter optimization. *ACM Transactions on Graphics 37*, 6 (Dec. 2018), 1– 14.

[ZJL20] Zhao S., Jakob W., Li T.-M.: Physics-based differentiable rendering: From theory to implementation. In *Proceedings of the ACM SIGGRAPH 2020 Courses, SIGGRAPH 2020* (New York, NY, USA, 2020), Association for Computing Machinery.

© 2022 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.