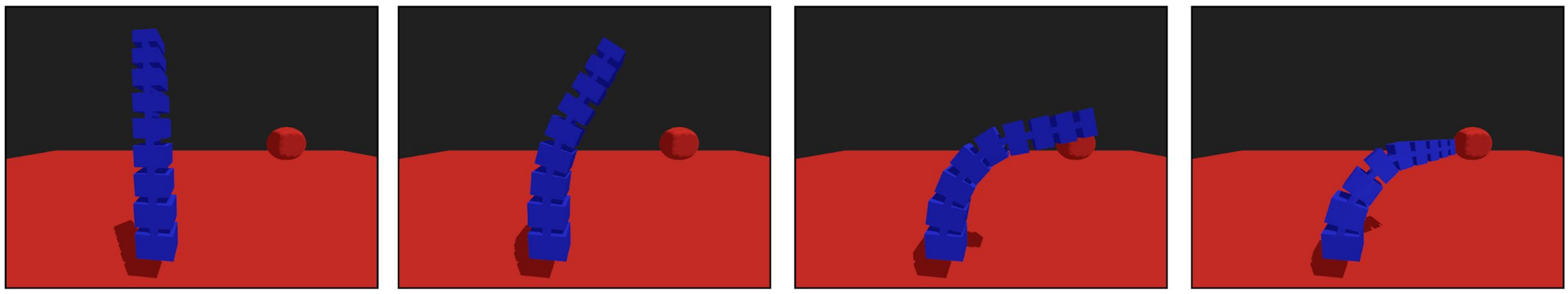
UNIVERSITY OF COPENHAGEN

DEPARTMENT OF COMPUTER SCIENCE

Animation-Based Soft Robot Control

Max Kragballe Nielsen

Abstract
Designing and implementing robot control programs is presently reserved for experts in fields such as Computer Science and Engineering. However, as robots are becoming an ever-growing part of our society, accessibility must increase. We propose an animation-based tool for crafting robot control programs, where a user can relay intent through key-frame drawings/renders using popular sculpting tools such as Blender and Maya.



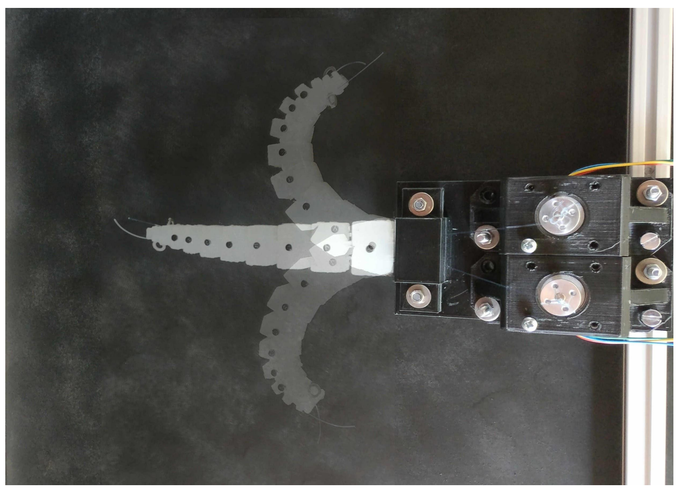
A user-authored soft robot trajectory, for which we want to learn corresponding control signals.

Introduction

In recent years, soft robotics has gained traction due to a rise in the availability of personal manufacturing.
Soft robots are well-suited for completing highly variable tasks, such as grasping, but as a consequence of their highly flexible frames, designing control programs is non-trivial.

Helping users design control programs has been proposed previously as in [1], where physical simulations were leveraged such that the design phase would only require simple instructions, like moving a point on the character, or specifying an angular momentum. In [2], animated plushies were designed to reenact idealized user-authored motions. Similarly to [1], a physical simulation was performed to ensure that the resulting control programs were realizable.

Range of motion.
A cable-driven soft robot actuated using stepper motors.



In our work, we aim to create an interface similar to the one presented in [2], but instead of requiring mesh information - such as connectivity and positions - we apply differentiable rendering to extract necessary information from user-provided keyframes. This provides an advantage, as input mesh quality becomes less of an issue.

Our approach is suited for problems such as,
- learning inverse kinematics
- aiding in the design of physically realistic animations

and we aim to extend the current work to solve problems such as object manipulation, where a user animates only how the object should behave.

Inverse Problem Statement

Given a user-authored trajectory,

$$\mathcal{T} = \left((I^{(0)}, I^{(1)}, \ldots, I^{(\tau)})\right)^T$$

we wish to find the control signals,

$$\mathbf{u}^* = \left((u^{(0)}, u^{(1)}, \ldots, u^{(\tau-1)})\right)^T$$

such that,

$$\mathbf{u}^* = \arg\min \mathcal{L}(\mathcal{T}, \hat{\mathcal{T}}(\mathbf{u}))$$

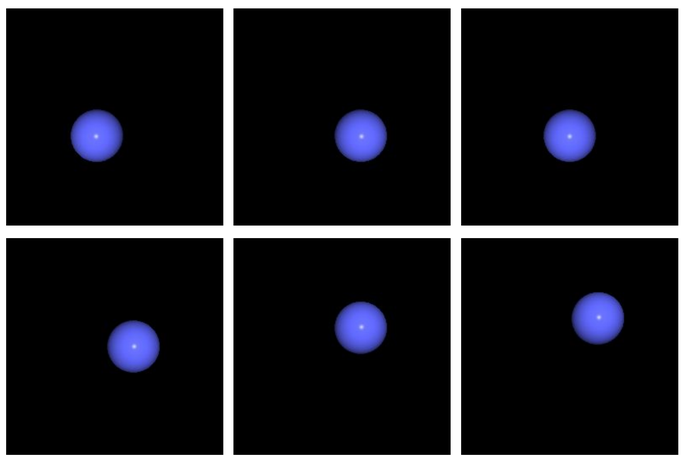
where $\hat{\mathcal{T}}(\mathbf{u})$ is the trajectory resulting from applying the control signals $\mathbf{u}$ to our soft robot.
To ensure that our control signals are realizable, the trajectory is generated through physical simulation.

Method

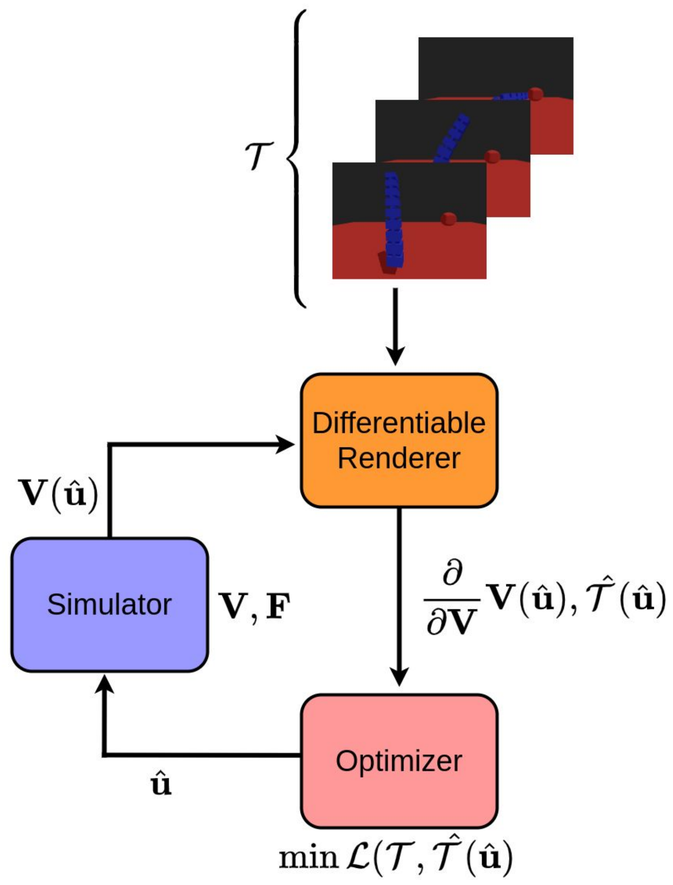
To process animated images directly, we make use of the DIB-R renderer presented in [3], which allows for gradients to be analytically computed for all pixels in an image. This enables us to perform gradient-based optimization over the control parameters, providing a speed-up in terms of convergence.

A common problem in soft robotics is that simulations diverge from reality, as a consequence of incorrect simulation parameter. This create a gap between simulations and the real world - the sim-2-real gap - that often leads to algorithms that work well in simulation, but fail in reality. To reduce this divergence, we use a differentiable simulator constructed similarly to the ones presented in [4], [5], which allows for faster identification of correct simulation parameters.

Example trajectory input.
A rigid sphere being moved around during a user-authored animation.



The workflow of our model.
The simulator generates a set of vertex positions for each time dependent control signal. The differentiable renderer then renders the model and computes positional derivatives using the user-authored input trajectory.
These derivatives are then used to minimize the trajectory loss by tuning the control parameters.



Future Work and Limitations

Currently, we are able to optimize object positions, orientations and velocities in simple physical systems. However, moving onwards to more difficult tasks such as grasping and locomotion we need to further develop our model, as well as our trajectory loss function (which is currently the loss function described for 3D object prediction in [3]).
Our robots are controlled through arduino-powered drivers, but our simulations assume that we have a system with no latency, as well as motors capable of a wide range of speeds. In reality, this is a pretty poor model of the motors, which has meant that we can only hope to transfer skills from tasks where speed is non-essential, such as moving an end-effector to a desired position.

A pneumatically actuated robot during actuation.

References

[1] Soft Body Locomotion, *Jie Tan, Greg Turk, and C. Karen Liu*, CM Trans. Graph. 31 4, Article 26 (July 2012)
[2] Interactive design of animated plushies, *James M. Bern, Kai-Hung Chang, and Stelian Coros*. ACM Trans. Graph. 36, 4, Article 80 (July 2017).
[3] Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer, *Wenzheng Chen, Jun Gao, Huan Ling, Edward J. Smith. Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler*, NeurIPS, 2019
[4] Augmenting Differentiable Simulators with Neural Networks to Close the Sim2Real Gap, *Heiden, E., Millard, D., Coumans, E., & Sukhatme, G. S.* (2020). arXiv preprint arXiv:2007.06045.
[5] Interactive differentiable simulation, *Heiden, E., Millard, D., Zhang, H., & Sukhatme, G. S.* (2019). arXiv preprint arXiv:1905.10706.