An End-to-End Differentiable Framework for Contact-Aware Robot Design (Supplementary Material)

Jie Xu[†], Tao Chen[†], Lara Zlokapa[†], Michael Foshey[†], Wojciech Matusik[†], Shinjiro Sueda[‡] and Pulkit Agrawal[†]
[†]Massachusetts Institute of Technology
[‡]Texas A&M University
http://diffhand.csail.mit.edu

I. Manipulator Grammar

While our work focuses on optimizing the continuous morphology parameters for a given robot design topology, we make the full pipeline as automatic as possible, from constructing the design all the way towards the optimization. Inspired by Zhao et al. [3], we represent the manipulator topology as a graph and construct a context-sensitive stochastic graph grammar for the manipulators to generate designs of different topologies and compute their articulated cage parameterization.

The grammar is shown in Figure 1. Each rule in the grammar can replace a non-terminal graph node with a sub-graph. More specifically, each rule contains a graph on its left hand side and a graph on its right hand side indicating the sub-graph before and after this rule derivation. Each circle with a letter is a nonterminal symbol and each one with model picture is a terminal symbol which is also the component from our constructed database. Specially, the circle with ϵ inside indicates it is an empty node which means that by this rule we are able to remove a nonterminal symbol "F" from the graph. To construct a manipulator, we start from the initial nonterminal symbol "S" and iteratively apply the rules to replace the nonterminal symbols in the graph until there is no nonterminal symbols in the graph. The grammar system also automatically compute the articulated cages during rule application.

With a small component database consisting of only five basic components, our grammar provides us with a expressive topology design space of various component combinations. The two manipulator configurations are picked among the designs in this grammar space.

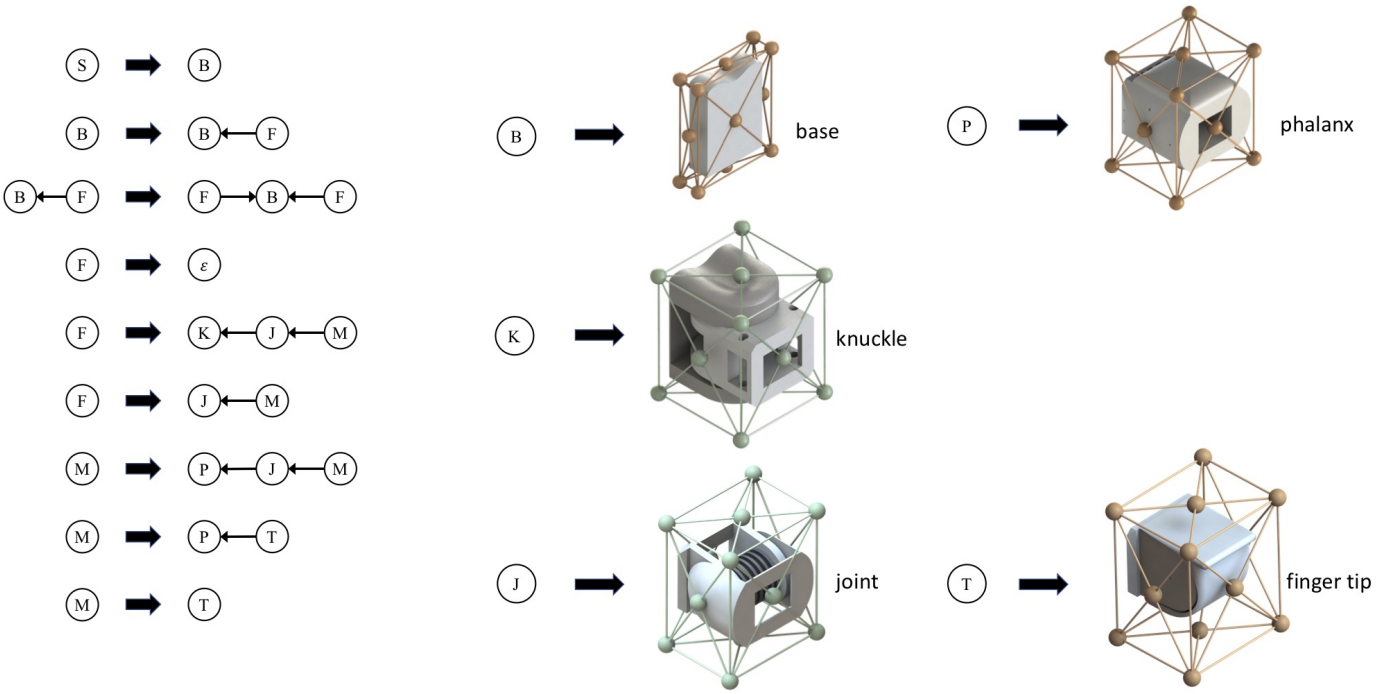

Fig. 1. **Context-sensitive stochastic graph grammar for manipulator.**

II. Experiment Details

In this section, we provide more details about experiments.

A. Tasks

We elaborate the designed tasks in this section including the task descriptions and the loss functions. We constructed five tasks in the experiments to test the performance of our method.

1) *Finger Reach*: In this task, the finger's base is mounted on the wall, and the finger is required to reach four scattered target positions in the space in sequence. The initial design of the finger is not long enough to reach two of them. Thus it requires the algorithm to optimize to finger to be longer in order to reach all four points. The cost $\mathcal{L}$ of this task is computed by:

$$\mathcal{L} = \sum_{t=1}^{T} c_u \|u_t\|^2 + c_p \|p_t - \hat{p}_t\| \tag{1}$$
$$\text{with } c_u = 0.1, c_p = 10$$

where $u_t \in [-1, 1]$ is the action at time t, p_t is the finger tip position at time t, and $\hat{p}_t$ is the target points at time t.

2) *Flip Box*: This task requires the finger to flip a heavy box by 90° and be as energy-efficient as possible. The bottom front edge of the heavy box is attached to the ground with a revolute joint. This task is more difficult than the previous one since the finger needs to interact with the box, which involves a rich amount of contacts and requires leverage of the contact force to flip the heavy box. The cost of this task is computed by:

$$\mathcal{L} = \sum_{t=1}^{T} c_u \|u_t\|^2 + c_{touch} \|p_t - p_{touch}\|^2 + c_{flip} \|\theta_t - \frac{\pi}{2}\|^2 \tag{2}$$
$$\text{with } c_u = 5, c_{touch} = \begin{cases} 1 & t < T/2 \\ 0 & t \geq T/2 \end{cases}, c_{flip} = 50 \tag{3}$$

where $u_t \in [-1, 1]$ is the action at time t, p_t is the finger tip position at time t, p_{tourch} is a point on the back surface of the box, and θ_t is the rotation angle of the box at time t. The second term is designed to encourage the manipulator to touch the box and provide some simple heuristics of solving the task.

3) *Rotate Rubik's Cube*: The finger is required to rotate the top layer of a Rubik's cube by 90°. The bottom of the Rubik's cube is fixed on the ground. In this task, there is no clear heuristics in the objective function to guide the finger to touch a specific place on the cube, so the finger needs to be optimized to find the correct strategy. The cost of this task is defined by:

$$\mathcal{L} = \sum_{t=1}^{T} \left(c_u \|u_t\|^2 + c_{touch} \|p_t - p_{cube}\|^2 \right) + c_{rotate} \|\theta_T - \frac{\pi}{2}\|^2 \tag{4}$$
$$\text{with } c_u = 5, c_{touch} = 0.1, c_{rotate} = 1000 \tag{5}$$

where u_t and p_t are same as previous tasks, p_{cube} is the center of the Rubik's cube, and θ_T is the rotation angle of the top layer of the cube at the last time step.

4) *Assemble*: In this task, two fingers need to collaborate together to push and insert a small box into its movable mount. The box and the hole on the mount have similar sizes, making the task much more challenging and requiring high-accuracy manipulation. Moreover, the movable mount needs to stay as close as possible to the original position to mimic a restricted working platform environment. The two fingers are mounted on a manipulator base that is allowed to move in the horizontal plane. The cost of this task is computed as:

$$\begin{aligned} \mathcal{L} = \sum_{t=1}^{T} & c_{mount} \|p_t^M - p_0^M\|^2 + c_{touch}(\|p_t^{left} - p_t^M\|^2 + \|p_t^{right} - p_t^{box}\|^2) \\ & + c_p \|p_t^{box} - p_{hole}\|^2 + c_{rotation} \|\theta_t^M - \theta_t^{box}\|^2 \\ & \text{with } c_{mount} = 15, c_{touch} = 1, c_p = 5, c_{rotation} = 50 \end{aligned} \tag{6}$$

where p_t^M is the position of the movable mount at time t, p_t^{left} and p_t^{right} are the finger tip positions of left finger and right finger at time t, p_t^{box} is the position of the small box at time t, θ_t^M and θ_t^{box} are the rotation angle of the mount and the box. The first term is used to penalize moving the mount too far away from the original place, the second term is designed to encourage the fingers to touch on the objects (but not indicate any specific position on the object), and the third term and the last term together is to measure how well the box is inserted into the mount.

5) *Free-form Gripper*: In this task, the algorithm needs to optimize the shape of two free-form gripper fingers in order to grasp a diamond-like object. The gripper fingers are cubes in the initial configuration but allowed for free-form deformation. We shake the object at the end of the episode to test the stability of the grasp. The control sequence in this task is pre-defined and fixed. The cost is computed by

$$\mathcal{L} = \sum_{t=1}^{T} h_t \tag{7}$$

where h_t is the height of the target object.

All the coefficients in the loss functions are selected to balance the importance and the value magnitude of different terms.

B. More Experiment Results

We provide the full optimization loss curves for *Assemble* task. In the *Assemble* task, we found the continuation method [1, 2] can further reduce the task cost with our method. More specifically, we scale down the contact force to provide the optimization a smoother function objective space, and then scale the contact force up to the true value. we set a curriculum for the contact scale in 3 stages. We set the contact scale to be 0.01 in the first stage, 0.1 in the second stage, and 1 in the third stage. In our method, we start the next stage as long as the convergence is detected by L-BFGS-B for the previous stage. Scaling down the contact force initially allows the optimization to find a good initial solution for the final harder task. To make a fair comparison with baselines, we trained all the baselines with the continuation methods as well. We split the entire 5000 training episodes into 3 stages evenly for the baselines. As shown in Figure 2, our method achieves significantly faster convergence and lower loss than all the baselines with or without the continuation method. While the continuation method speeds up the optimization and leads to lower converged loss using our method, it does not help improve the baselines' optimization performance. It even slows down the optimization in the case of CMA-ES. We hypothesize that it is because the baselines cannot solve the task (even the easiest *Assemble* task with contact scale 0.01) within a small number of iterations. Hence, the curriculum does not provide any benefit to the optimization.

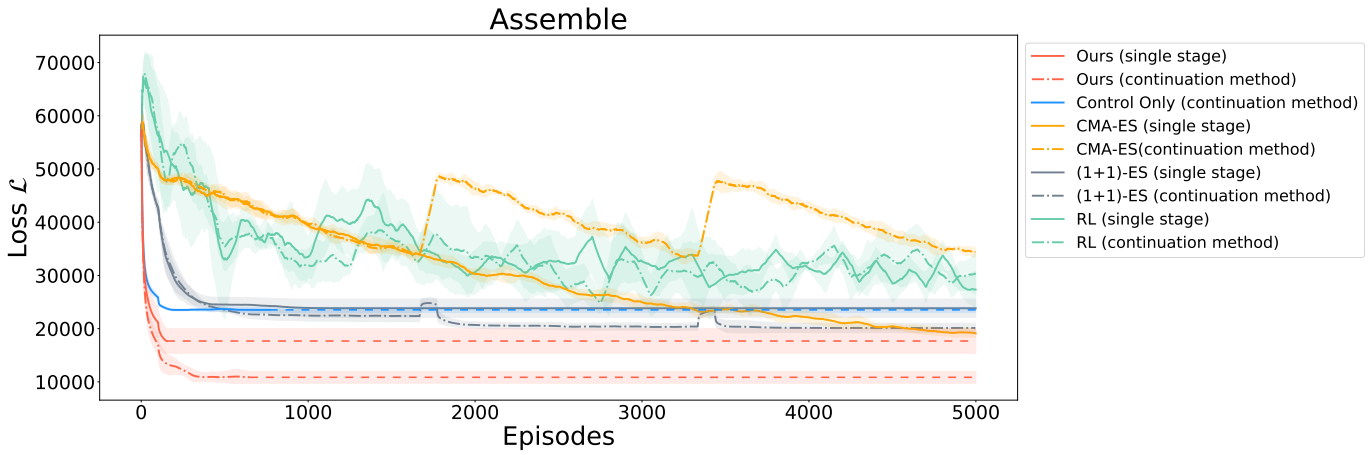

Fig. 2. **Full plot of comparison curves for *Assemble* task**. The curves are smoothed for better visualization and comparison.

We also show the loss curves of our deformation-based parameterization method and mesh-based parameterization method on the *Free-form Gripper* task. We run each method for 30 independent times with random initial morphology guesses and plot the average optimization loss curve in Figure 3. Although our method fails in one optimization trial and the mesh parameterization succeeds in grasping the object in all 30 optimization trials, we can see that our cage-based parameterization leads to better average performance. This is because our reduced morphology parameter space allows the optimization to explore more easily than the high DoF parameter space provided by the mesh parameterization method.

REFERENCES

[1] Eugene L Allgower and Kurt Georg. *Introduction to numerical continuation methods*. SIAM, 2003.

[2] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[3] Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

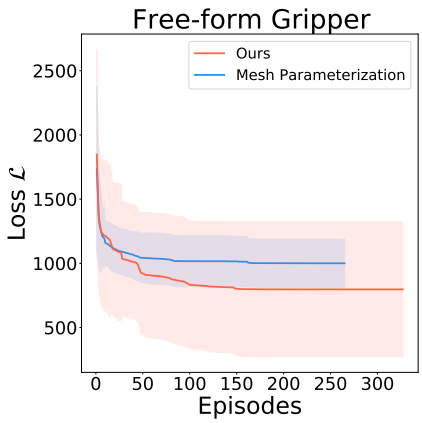

Fig. 3. **Optimization curve comparison for *Free-form Gripper* task**. The horizontal axis is the number of simulation episodes during optimization, and the vertical axis is the loss value. The experiment results are averaged from 30 independent optimization runs with different initial guesses.