Contact Sampling for Contact-Rich Manipulation

Spring 2023 Supervised Research Report

Shao Yuan Chew Chia
Harvard University
Cambridge, MA
shaoyuan_chewchia@college.harvard.edu

H.J. Terry Suh
MIT
Cambridge, MA
hjsuh94@mit.edu

Sizhe Li
MIT
Cambridge, MA
sizheli@mit.edu

Tao Pang
Boston Dynamics AI Institute
Cambridge, MA
tpang@theaiinstitute.com

***Abstract*—Motion planners for contact-rich manipulation are limited by current implementations of contact samplers. In this report we present initial progress towards developing a generalizable and performant goal-conditioned contact sampler. We explore the use of randomized smoothing and analytic smoothing to overcome the flatness of the cost landscape under exact dynamics, thereby enabling the use of gradient based methods to improve sampled configurations. We also show the effects of different smoothing and gradient descent parameters on the cost landscape and gradient descent trajectories on simple planar pushing examples.**

I. Introduction

Contact-rich manipulation is a class of robotic tasks that involve close and complex interaction between the robot and its environment [1]. This is an important problem in robotics because it can unlock the ability of robots to efficiently perform dexterous physical tasks such as wiping [2], molding [3], [4] and reorienting [5], which humans perform instinctively but thus far has remained challenging for robots. Planning for such tasks is challenging because of the discontinuous, non-smooth and high-dimensional contact dynamics. Recent work has identified contact sampling (figuring out where and how a robot should make contact with an object in order to achieve a goal configuration) as an important bottleneck in the effectiveness of model-based methods for contact-rich manipulation [4], [6].

In this report we present initial work on developing a generalizable and performant goal-conditioned contact sampler.

In [6], Pang et al. demonstrate the potential of sampling based planners for contact rich manipulation. They use smoothing of contact dynamics to avoid explicit enumeration of contact modes and highlight the importance of the contact sampling step in increasing state space coverage of the RRT. However, they use a naive implementations of the contact sampler. For the planar pusher and planar hand tasks they randomly sample a single contact configuration on the object's surface. For the more complex manipulator of the Allegro hand, they use Eigen grasps [7] to pick a random direction in the hand configuration space and then simulate closing the hand along that direction until contact is established. These methods are used to find *any* contact configuration rather than a good contact configuration conditioned on the goal. Another shortcoming of their implementation is that the contact sampler needs to be hand designed for the particular manipulator-object pair.

In the context of deformable object manipulation, work by Li et al. [4] demonstrates how contact sampling can be used together with trajectory optimization to complete multi-stage Plasticine shaping tasks which require making and breaking contact with the Plasticine. They first use optimal transport to define a heuristic for which object particles have to move the greatest distance to achieve the goal configuration of the object. They then manually specify how the end-effector should be placed in relation to those particles. However, since their heuristic is object-centric, it does not easily inform how a complex end-effector such as a dexterous hand should make contact with the object.

In our work, we hope to develop a contact sampler that is highly general, working across manipulators (from simple to complex) and objects, whether rigid or deformable.

Unlike previous work on grasp synthesis [8], [9], we do not have the explicit goal of grasp stability. Additionally, we do not restrict the contact points to be at the fingertips of the manipulator, allowing for contact between the object and manipulator to be made anywhere along the surface of the manipulator.

Our approach is also different from functional grasp synthesis [10], [11], where the aim is to grasp the object or tool in a way such that it can be used to perform a particular function, for example, grasping a mug such that it can be *placed* somewhere or such that its contents can be *poured* into another container [10], [11]. Functional grasp synthesis encodes higher level constraints on how to use an object or tool for a particular function. Our problem is more low-level and involves manipulating the object itself into a desired shape or pose. Additionally, these works do not deal with non-prehensile manipulation, which is an important mode of manipulation for achieving extrinsic dexterity.

II. Problem Formulation

Given a goal state x_g and current state x of the object, we want to find the robot configuration q, that would, in one subsequent command u, minimize the object's distance to the goal,

$$\min_{q \in \mathcal{A}(x)} d(x_g, f(x, q, u)). \tag{1}$$

This is a short horizon version of the longer horizon question: Given a goal state x_g and current state x of the object, how should you position the robot q to best reach x_g, through subsequent commands u.

Note that embedded within this formulation is the question of what singular or sequence of u's should be taken in order to achieve the minimum distance. In Sec. IV, where we explore randomized smoothing of the cost landscape, we minimize the expected cost over Gaussian samples of one step u's. In Sec. V where we explore analytic smoothing of contact dynamics, we minimize the cost under the nominal position command $\bar{u} = \bar{q}$ of the current position of the robot (no movement).

We denote the full state of the system s as being composed of the unactuated object states x and actuated robot states q,

$$s = \begin{bmatrix} x \\ q \end{bmatrix}. \tag{2}$$

In this report we will consider a discrete-time system with quasistatic dynamics where the control input is an absolute position command u.

$$s' = f(x, q, u) \tag{3}$$

The ideas discussed should extend to fully dynamic settings, but we will leave those modifications to future work.

III. Solution Sketch

Alg. 1 describes the outline of the contact sampling method we propose. In this report we only discuss two formulations for determining the gradient at a sample point, but we have made the code base modular to facilitate experimentation with different gradient descent algorithms, cost functions and dynamical systems as well.

Algorithm 1: Contact Sampling

```
1 Input: Initial object state $x$, goal object state $x_g$;
2 Output: Best robot states $\{q_n\}_{n=0}^{N-1}$;
3 $\{q_{init_m}\}_{m=0}^{M-1}$ ← Rejection sample initial admissible
    positions from uniform distribution over workspace;
    $\{q_{final_m}\}_{m=0}^{M-1} = \texttt{GradientDescent}(\{q_{init_m}\}_{m=0}^{M-1})$ ;
4 $\{q_n\}_{n=0}^{N-1}$ ← Select $N$ lowest cost $\{q_{final_m}\}_{m=0}^{M-1}$;
5 return $\{q_n\}_{n=0}^{N-1}$
```

The key challenge in applying such a gradient based method to find the solution to (1) is that the gradients are zero in the absence of contact, making it a difficult landscape to optimize over.

IV. Randomized Smoothing of the Cost

The first approach we will use to address the flat cost landscape is randomized smoothing of the cost. Instead of minimizing the exact cost (1), we minimize the expectation of the cost at a given point when we inject a Gaussian perturbation in position command u,

$$\min_{q \in \mathcal{A}(x)} \mathbb{E}_w \left[d(x_g, f(x, q, u = q + w)) \right] \qquad w \sim \mathcal{N}(0, \sigma^2). \tag{4}$$

A. Cost Landscape Computation

To generate a visualization of this smoothed cost landscape, we first discretize the state space into a grid. At each point on the grid we take samples of actions which are normally distributed and centered on the position of the grid point. We calculate the cost based on the state of the object after rolling out exact dynamics for each action sample. We then take the mean of the rolled out costs, corresponding to the zeroth order gradient estimation described in Sec. IV-B.

B. Gradient Computation

To use zeroth-order methods to calculate the gradient, we choose a surrogate objective to (4),

$$\min_q \mathbb{E}_{v,w} \left[d(x_g, f(x, q+v, q+v+w)) \right], \tag{5}$$

where $w \sim \mathcal{N}(0, \sigma_w^2 \mathbf{I})$ and $v \sim \mathcal{N}(0, \sigma_v^2 \mathbf{I})$.

Using Stein's lemma, the gradient of (5) can be written as

$$\begin{aligned} &\nabla_q \mathbb{E}_{v,w} \left[d(x_g, f(x, q+v, q+v+w)) \right] \\ &= \mathbb{E}_{v,w} \left[\frac{v}{\sigma_v^2} \left[d(f(x, q+v, q+v+w), x_g) - \mu^* \right] \right], \end{aligned} \tag{6}$$

where $\mu^* = E_{v,w} \left[d(x_g, f(x, q+v, q+v+w)) \right]$ which has the unbiased estimator of the sample mean,

$$\mu_\rho(q) = \frac{1}{N} \sum_{i=1}^{N} \left[d(f(x, q+v_i, q+v_i+w_i), x_g) \right]. \tag{7}$$

Thus, the natural estimator of the gradient (6) is

$$\frac{1}{\sigma_v^2} \frac{1}{N} \sum_{i=1}^{N} v_i \left[d(f(x, q+v_i, q+v_i+w_i), x_g) - \mu_\rho(q) \right]. \tag{8}$$

C. Results and Discussion

1) Effect of Action Standard Deviation on Cost Landscape: A larger action standard deviation, σ_u, results in greater smoothing of the cost landscape which can also be interpreted as allowing a sample point to pull in information from further away at the cost of introducing bias.

Fig. 1 (right) depicts a situation where σ_u is high relative to the magnitude of the action required to push the object into the goal configuration. The object is very close to the goal, thus for $\sigma_u = 0.1$ all positions that make contact with the object are higher cost than positions further away from the object. This is because random actions taken from each of the positions in contact would on average push the box further away from the goal. By reducing σ_u to 0.01, what was previously a hill becomes a very slight valley along the left face of the object (Fig. 2 (left)). Reducing σ_u also has the effect of making the cost landscape flatter in general as can be seen from Fig. 2 and 3.

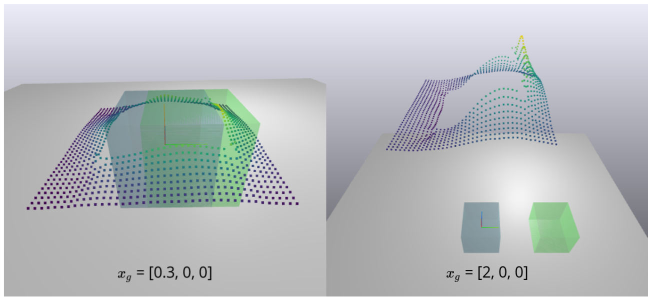

Figure 1: Effect of distance between initial and goal object positions on cost landscapes with randomized smoothing where action standard deviation $\sigma_u = 0.1$

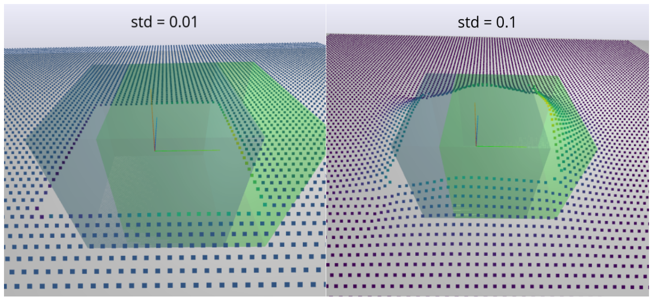

Figure 2: Effect of action standard deviation σ_u on cost landscapes with randomized smoothing for $x_g = [0.3, 0, 0]$.

2) Effect of Aggregation Functions on Cost Landscape: In Fig. 4, we see that taking the *Min* of the costs of the rolled out action samples creates a landscape that is flat everywhere that the action samples were not able to decrease the cost, a has a sinkhole where the cost was improved by the action samples with a global minimum at the center of the left face of the box. To do gradient descent on this cost landscape the gradient computation described in Sec. IV-B would need to be reformulated. It may be worth exploring as it seems that the region of attraction to the global minimum is larger for *Min* as compared to *Mean*. Additionally, the gradient descent algorithm could be sped up by discarding samples that do not move after a few iterations. It would not make sense to do gradient descent on the max cost landscape (right), but it was included just for comparison to build intuition.

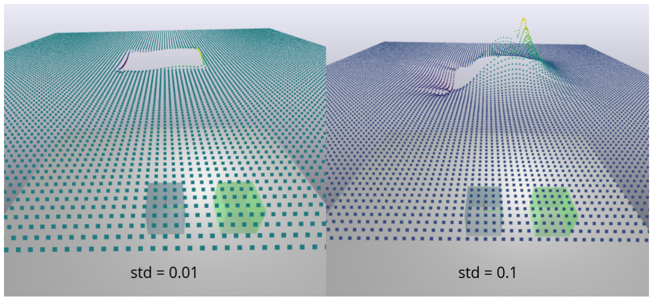

Figure 3: Effect of action standard deviation σ_u on cost landscapes with randomized smoothing for $x_g = [2, 0, 0]$.

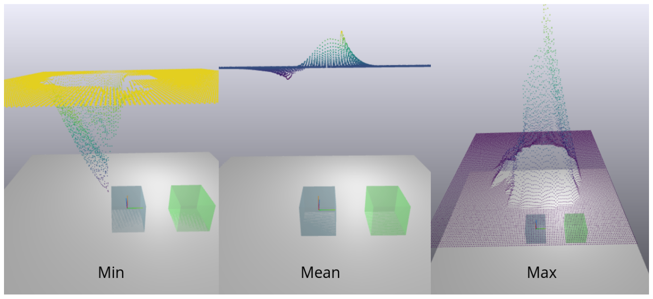

Figure 4: Effect of different aggregation functions on cost landscapes with randomized smoothing for $x_g = [2, 0, 0]$.

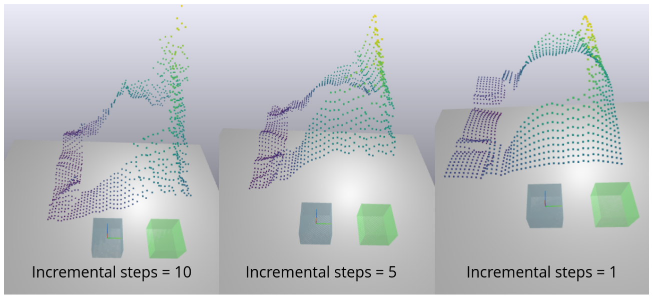

Figure 5: Effect of number of incremental rollout steps on cost landscapes with randomized smoothing for $x_g = [2, 0, 0]$, $\sigma_u = 0.5$. For example, for steps = 10, the absolute position command would be discretized into and rolled out over 10 steps, such that the difference between the robot's current position and the commanded position at each intermediate step would be smaller.

3) Non-physical Effects of Convex Relaxation of Contact Dynamics on Cost Landscape: Anitescu's convex relaxation of contact dynamics introduces non-physical effects [12] which are magnified when taking a larger action in a single step as can be seen in in Fig. 5. The tangential velocity between the robot and object creates phantom forces from a distance. We hypothesize that this greater coupling between the position of the robot and the object results in the increased smoothness we see when rolling out the dynamics over 1 step instead of 10 steps.

4) Performance of contact sampling: Fig. 6 shows that performing gradient descent with the zeroth order batch gradient was able to shift the initially sampled points to lower cost positions. A significant number of points "run away" from the object (in particular, points initially along the left, bottom and lower part of the right face of the object). There are three minima at the top left, top right and bottom right of the box that we see some points successfully make their way to, and other points head towards.

5) Effect of Perturbation Standard Deviation on Gradient Descent Trajectories: Comparing Fig. 8 and 7 we see that if σ_u is large relative to σ_q, the trajectories of the points are more jagged and larger steps are taken (Fig. 8 (right) and Fig. 7 (left)). If σ_u is small relative to σ_q, the trajectories of the

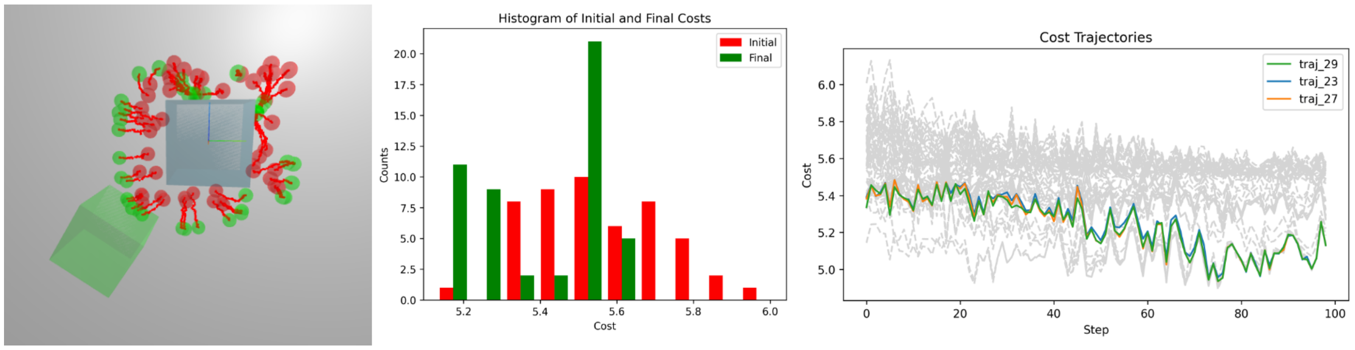

Figure 6: Results of gradient descent with zeroth order batch gradient on cost with randomized smoothing for $x_g = [-1.5, -1.5, 1]$, $\sigma_q = 0.1$, $\sigma_u = 0.1$, $h = 0.01$. **Left**: visualization of gradient descent iterations with initial iteration in red, and final iteration in green; **middle**: histogram of initial and final costs; **right**: trajectories of cost through gradient descent iterations with the 3 trajectories with the lowest final cost in color and the rest of the trajectories in grey.

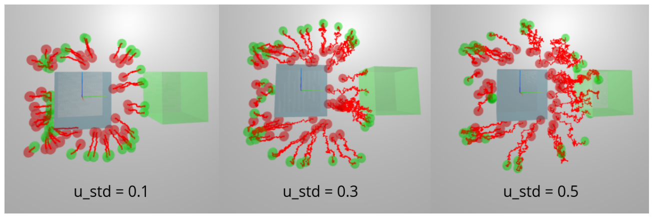

Figure 7: Effect of action standard deviation σ_u on gradient descent with zeroth order batch gradient on cost with randomized smoothing for $x_g = [2, 0, 0]$, $\sigma_q = 0.1$, $h = 0.01$.

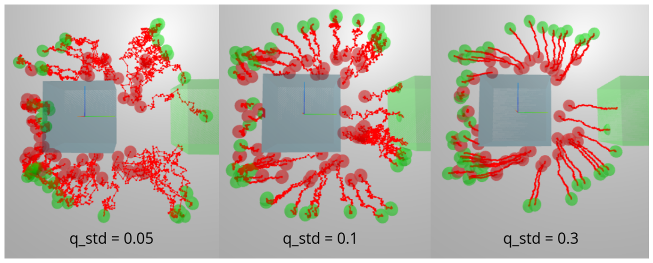

Figure 8: Effect of position standard deviation σ_q on gradient descent with zeroth order batch gradient on cost with randomized smoothing for $x_g = [2, 0, 0]$, $\sigma_u = 0.3$, $h = 0.01$.

points are straighter and smaller steps are taken (Fig. 8 (left) and Fig. 7 (right)).

In Fig. 8 (right), we see that no points end up close to the surface of the object, even on the left face of the object. We see that when σ_q is too large, it has the effect of creating a buffer zone around the object which is undesirable for our use case.

To explain why points in Fig. 7 (left) end up spread out along the left face of the object while points in 7 (middle) and (right) end up in the center of the left face, we note the correspondence between the gradient descent results in Fig. 7 and the cost landscape of Fig. 2. The gradient towards the center of the left face is steeper for larger σ_u.

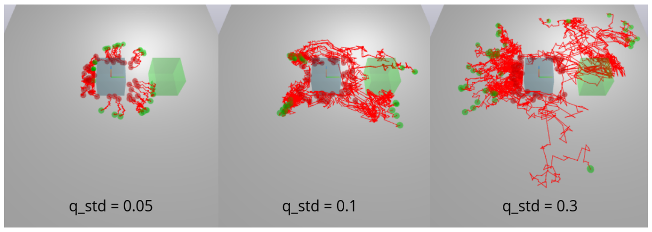

Figure 9: Effect of position standard deviation σ_q on gradient descent with zeroth order batch gradient on cost with randomized smoothing for $x_g = [2, 0, 0]$, $\sigma_u = 0.3$, $h = 0.05$.

6) Effect of Step Size on Gradient Descent Trajectories: Contrasting Fig. 8 ($h = 0.01$) and 9 ($h = 0.05$), we see the importance of tuning the step size h based on the σ's used. Using a large step size coupled with large σ as shown in Fig. 9 (right), results in chaotic trajectories that do not converge near the surface of the object.

V. Analytic Smoothing of Contact Dynamics

Instead of doing randomized smoothing the cost directly as we did in Sec. IV, we can instead choose to analytically smooth the dynamics of the system, as implemented in [6], and evaluate the same cost. This is because analytic smoothing of the cost is significantly harder than the dynamics which is more structured.

A. Cost Landscape Computation

To generate a visualization of this smoothed cost landscape, we similarly first discretize the state space into a grid. However, different from the randomized smoothing cost landscape described in Sec. IV-A, at each point on the grid we do not take samples of actions. Instead, we calculate the cost based on the state of the object after rolling out the smoothed dynamics for the nominal action $d(x_g, f_x^\rho(x, q, u = q))$.

B. Gradient Computation

We first introduce the notation for the analytically smoothed dynamics of the object as

$$x' = f^{\rho}_{x}(x, q, u), \tag{9}$$

where the subscript x indicates that we are extracting only the next state of the object instead of the full state of the system 2, and the superscript ρ indicates that the dynamics are smoothed.

We denote the absolute position command u as a function of the current robot position q and a relative position command u_r

$$u(q, u_r) := q + u_r. \tag{10}$$

We then define an alternate dynamics function that takes the relative command as input instead of the absolute command,

$$f^{\rho r}_{x}(x, q, u_r) = f^{\rho}_{x}(x, q, u(q, u_r)). \tag{11}$$

In order to perform gradient descent we'd like to calculate the gradient of the cost with respect to the position of the robot q,

$$\begin{aligned}&\nabla_q d(f^{\rho r}_{x}(x, q, u_r), x_g)\\ &= \frac{\partial}{\partial q} x'^{\top}\bigg|_{x'=f^{\rho r}_{x}(\bar{x},\bar{q},\bar{u}_r)} \frac{\partial}{\partial x'} d(x_g, x')\bigg|_{x'=f^{\rho r}_{x}(\bar{x},\bar{q},\bar{u}_r)},\end{aligned} \tag{12}$$

where we take $u_r = 0$ or equivalently $u = q$.

The first term of (12) can be found using the chain rule for total derivatives,

$$\begin{aligned}\frac{\partial}{\partial q} x' &= \frac{\partial}{\partial q} f^{\rho r}_{x}(x, q, u_r)\bigg|_{\bar{x},\bar{q},\bar{u}_r}\\ &= \frac{\partial}{\partial q} f^{\rho}_{x}(x, q, u(q, u_r))\bigg|_{\bar{x},\bar{q},\bar{u}=\bar{u}_r+\bar{q}}\\ &= \frac{\partial}{\partial q} f^{\rho}_{x}(x, q, u(q, u_r))\bigg|_{\bar{x},\bar{q},\bar{u}=\bar{u}_r+\bar{q}}\\ &\qquad + \frac{\partial}{\partial u} f^{\rho}_{x}(x, q, u(q, u_r))\bigg|_{\bar{x},\bar{q},\bar{u}=\bar{u}_r+\bar{q}} \frac{\partial u}{\partial q}\\ &= \mathbf{A}_{\mathtt{xq}}(\bar{x}, \bar{q}, \bar{u}_r + \bar{q}) + \mathbf{B}_{\mathtt{x}}(\bar{x}, \bar{q}, \bar{u}_r + \bar{q}),\end{aligned} \tag{13}$$

where we note that $\partial u/\partial q = \mathbf{I}$ for our choice of $\bar{u}_r = 0$.

And if we are considering the specific cost function $d(x_g, x')$ of weighted l_2-norm cost,

$$d(x_g, x') = (x_g - x)^{\top}\mathbf{Q}(x_g - x'), \tag{14}$$

the gradient of (14) with respect to x' is

$$\frac{\partial}{\partial x'} d(x_g, x') = -2\mathbf{Q}(x_g - x'). \tag{15}$$

Substituting (13) and (13) into (12) we get

$$\begin{aligned}&\nabla_q d(f^{\rho r}_{x}(x, q, u_r), x_g)\\ &= -2[\mathbf{A}_{\mathtt{xq}}(\bar{x}, \bar{q}, \bar{u}_r + \bar{q}) + \mathbf{B}_{\mathtt{x}}(\bar{q}, \bar{q}, \bar{u}_r + \bar{q})]\mathbf{Q}(x_g - x'),\end{aligned} \tag{16}$$

which is the final form of the equation we use to calculate the gradient.

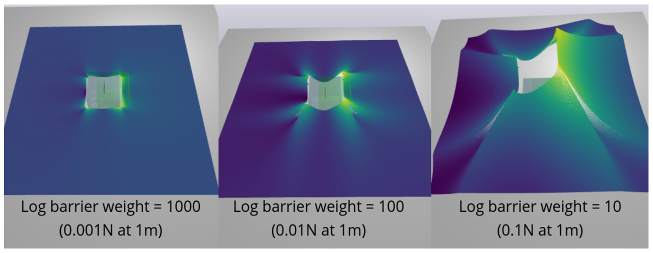

Figure 10: Effect of log barrier weight κ on cost landscapes with analytically smoothed contact dynamics for $x_g = [0.3, 0, 0]$.

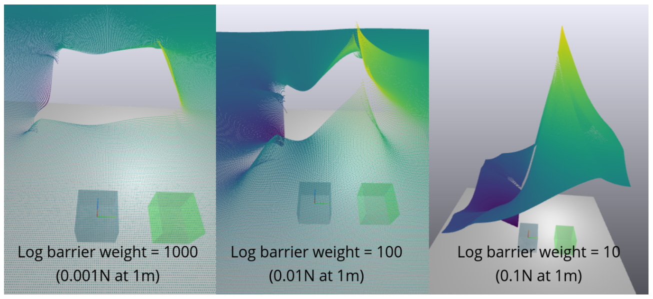

Figure 11: Effect of log barrier weight κ on cost landscapes with analytically smoothed contact dynamics for $x_g = [2, 0, 0]$.

C. Results and Discussion

1) Effect of Log Barrier Weight on Cost Landscape and Gradient Descent Trajectories: Similar to the effect of increasing σ in the context of zeroth order gradient estimation, reducing the log barrier weight κ, which corresponds to an increased the amount of smoothing, also makes the cost landscape more contoured, increasing the height of the peaks and depths of the valleys (Fig. 10-12).

Having κ be too small relative to the distance between the object's current position and goal position, as shown in Fig. 16 (right), results in final positions of the samples that are away from the surface of the object. The large force at a distance means that if the robot were to be closer to the surface it would exert too strong a force on the object and would end up increasing the cost. This is similar to what we see in Fig. 1 where the goal position of the object being too close to the initial position results in the cost of all points along the surface of the object being higher cost than the cost further away.

VI. Comparison between Randomized Smoothing of the Cost and Analytic Smoothing of the Contact Dynamics

We can see that analytic smoothing of contact dynamics leads to a smoother cost landscape than randomized smoothing of the cost directly. This translates to smoother trajectories of the sampled points during gradient descent as seen in Fig. 13-16 as compared to Fig. 6-9.

In the examples explored in this initial work, optimizing over the cost landscape created by the analytically smoothed contact dynamics seems to result in a greater ability for

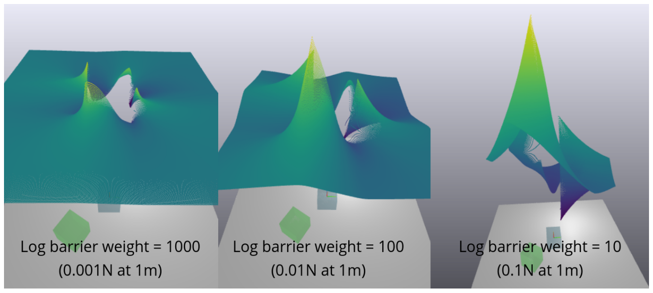

Figure 12: Effect of log barrier weight κ on cost landscapes with analytically smoothed contact dynamics for $x_g = [-1.5, -1.5, 1]$.

sampled points to travel towards at least locally optimal positions (Fig. 13), as opposed to "running away" from the object (Fig. 6).

We suspect that these differences arise from smoothing of the cost landscape vs smoothing of the dynamics, as opposed to randomized smoothing vs analytic smoothing or zeroth order gradient estimation vs first order gradient estimation, but more experiments will need to be carried out to confirm this suspicion.

A. *Computation Time*

Gradient descent with zeroth order batch gradients on the cost landscape with randomized smoothing, as described in Sec. IV-B, takes approximately 11 minutes for 100 iterations on 50 initial points, with 100 pairs of q and u perturbations. Fig. 6-9 show experiments run with these parameters. The time complexity is $\mathcal{O}(n_{\text{samples}} \cdot n_{\text{perturbations}} \cdot n_{\text{increments}} \cdot n_{\text{iterations}}/n_{\text{threads}})$, where n_{samples} is the number of initial sample points (Alg. 1), $n_{\text{perturbations}}$ is the number of pairs of position and action perturbations (8), $n_{\text{increments}}$ is the number of incremental steps the dynamics are rolled out over (Sec. IV-C3), and n_{threads} is the number of parallel instances used to roll out the dynamics.

Gradient descent for analytic smoothing of contact dynamics takes approximately 17.5 seconds for 700 iterations on 50 initial points, (as shown in Fig. 13-16). The time complexity is $\mathcal{O}(n_{\text{samples}} \cdot n_{\text{iterations}}/n_{\text{threads}})$.

All experiments were run on a Linux machine with 31.3 GB of RAM, with an Intel Core i7-6700 CPU @ 3.40GHz x 8 processor.

VII. Conclusion and Future Work

This report has described some initial progress towards creating a generalizable goal-conditioned contact sampler. In this section we lay out our roadmap for continued development of the method.

1) **Implement a metric for comparing performance of different parameters and methods**. In our first set of experiments, we collected the mean final cost of all samples as well as the mean final cost of the n-best samples in an attempt to compare performance between runs. While this can be used to compare the performance of different parameters within the same method (randomized smoothing of cost or analytic smoothing of dynamics), it cannot be used to compare across different methods (since the cost means different things in the different methods). Furthermore, it does not penalize the sample points being far away from the object's surface. We feel a better performance metric would be the n-best cost after a one-step trajectory optimization (calculated using least-squares) to find the optimal u_r^*, capped at a max u_r, on the analytically smoothed dynamics. We believe this is a good metric as it closely parallels the way the contact sampler will be used within RRT where we would like to find a robot configuration that will get us as far as possible towards a sub-goal.
2) **Evaluate performance impact of non-physical effects**. Using the new metric described above, determine whether incremental rollouts of the dynamics is truly necessary as it significantly increases runtime of the zeroth order randomized smoothing method.
3) **Rescaling the cost landscapes**. We'd like to recompute the cost landscapes shown in this report but with the cost values normalized to allow differences other than scale caused by changing the parameters to become more apparent.
4) **Try a variance/log barrier weight schedule**. This could help with having the final positions actually be in contact with the object, while speeding up convergence.
5) **Try using the Torch optimizer over our vanilla gradient descent implementation**.
6) **Explore gradient estimation based on min instead of mean**. As described in Sec. IV-C2 this could speed up runtimes and improve the quality of the contact samples.
7) **Implement the method for more complex rigid body manipulators**.
8) **Implement the method for deformable objects**.
9) **Integrate the contact sampling method into RRT**.
10) **Integrate the contact sampling method into trajectory optimization**.

While our initial examples simple, and significant work remains to be done, our early experiments make us optimistic that we'll be able to develop a practical and effective method for contact sampling. We are excited to see the new capabilities that this will enable in contact-rich manipulation planners!

References

[1] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba, "A review on reinforcement learning for contact-rich robotic manipulation tasks", *Robotics and Computer-Integrated Manufacturing*, vol. 81, p. 102 517, Jun. 1, 2023, ISSN: 0736-5845. DOI: 10.1016/j.rcim.2022.102517. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584522001995 (visited on 03/11/2023).

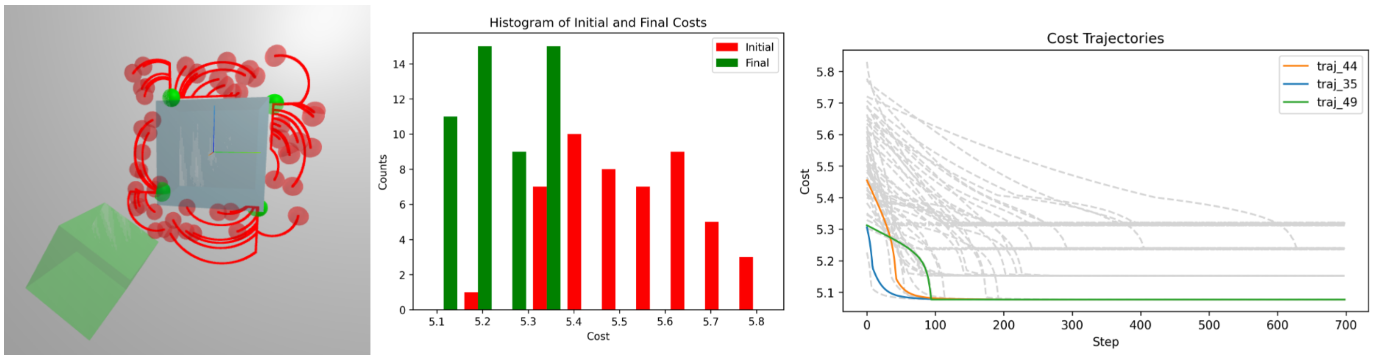

Figure 13: Gradient descent iterations for first order batch gradients $x_g = [2, 0, 0]$, $\kappa = 100$, $h = 0.01$.

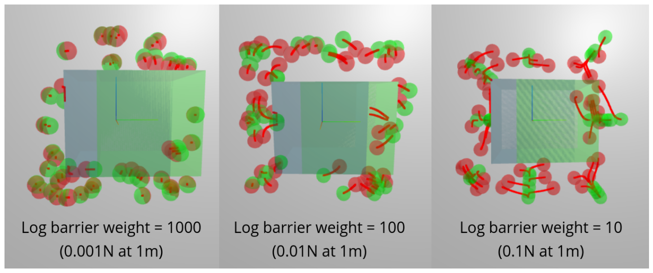

Figure 14: Effect of log barrier weight κ on gradient descent with first order batch gradients and analytically smoothed contact dynamics for $x_g = [0.3, 0, 0]$.

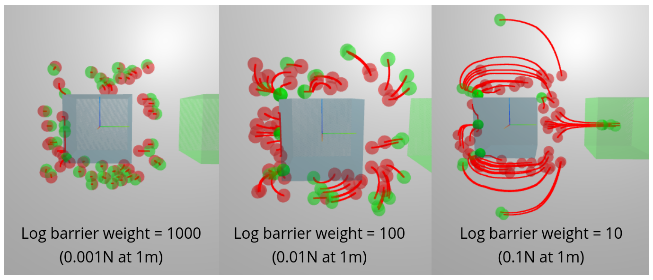

Figure 15: Effect of log barrier weight κ on gradient descent with first order batch gradients and analytically smoothed contact dynamics for $x_g = [2, 0, 0]$.

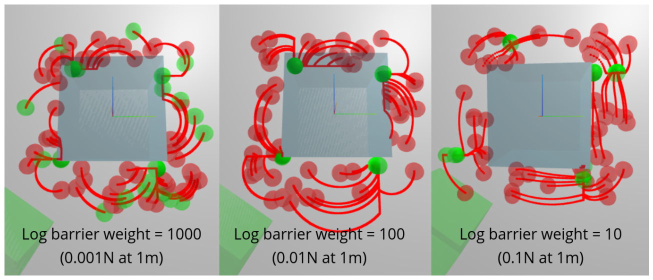

Figure 16: Effect of log barrier weight κ on gradient descent with first order batch gradients and analytically smoothed contact dynamics for $x_g = [-1.5, -1.5, 1]$.

[2] M. V. Balakuntala, U. Kaur, X. Ma, J. Wachs, and R. M. Voyles. “Learning Multimodal Contact-Rich Skills from Demonstrations Without Reward Engineering”. Comment: Accepted at IEEE ICRA 2021. This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible. arXiv: 2103.01296 [cs]. (Apr. 14, 2021), [Online]. Available: http://arxiv.org/abs/2103.01296 (visited on 05/23/2023), preprint.

[3] J. Ondras, D. Ni, X. Deng, Z. Gu, H. Zheng, and T. Bhattacharjee, “Robotic Dough Shaping”, in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, Nov. 2022, pp. 300–307. DOI: 10.23919/ICCAS55662.2022.10003767.

[4] S. Li, Z. Huang, T. Du, H. Su, J. B. Tenenbaum, and C. Gan. “Contact Points Discovery for Soft-Body Manipulations with Differentiable Physics”. Comment: ICLR 2022 Spotlight. Project page: http://cpdeform.csail.mit.edu. arXiv: 2205.02835 [cs]. (May 5, 2022), [Online]. Available: http://arxiv.org/abs/2205.02835 (visited on 03/06/2023), preprint.

[5] J. Xu, T.-K. J. Koo, and Z. Li, “Sampling-based finger gaits planning for multifingered robotic hand”, *Autonomous Robots*, vol. 28, no. 4, pp. 385–402, May 1, 2010, ISSN: 1573-7527. DOI: 10.1007/s10514-009-9164-5. [Online]. Available: https://doi.org/10.1007/s10514-009-9164-5 (visited on 04/25/2023).

[6] T. Pang, H. J. T. Suh, L. Yang, and R. Tedrake. “Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-dynamic Contact Models”. Comment: The first two authors contributed equally to this work. arXiv: 2206.10787 [cs]. (Feb. 27, 2023), [Online]. Available: http://arxiv.org/abs/2206.10787 (visited on 03/15/2023), preprint.

[7] M. Ciocarlie, C. Goldfeder, and P. Allen, “Dexterous Grasping via Eigengrasps: A Low-dimensional Approach to a High-complexity Problem”, *Robotics: Science and systems manipulation workshop-sensing and adapting to the real world*,

[8] S. Chen, A. Wu, and C. K. Liu. "Synthesize Dexterous Nonprehensile Pregrasp for Ungraspable Objects". Comment: 11 pages, 9 figures, SIGGRAPH Conference Proceedings 2023. arXiv: 2305.04654 [cs]. (May 8, 2023), [Online]. Available: http://arxiv.org/abs/2305.04654 (visited on 05/22/2023), preprint.

[9] Y. Fan, X. Zhu, and M. Tomizuka, "Optimization Model for Planning Precision Grasps with Multi-Fingered Hands", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 1548–1554. DOI: 10 . 1109 / IROS40897 . 2019 . 8967560.

[10] H. Dang and P. K. Allen, "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 1311–1317, ISBN: 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1. DOI: 10.1109/IROS.2012.6385563. [Online]. Available: http://ieeexplore.ieee.org/document/6385563/ (visited on 04/24/2023).

[11] P. Mandikal and K. Grauman. "Learning Dexterous Grasping with Object-Centric Visual Affordances". Comment: Accepted at ICRA 2021. arXiv: 2009.01439 [cs]. (Jun. 16, 2021), [Online]. Available: http://arxiv.org/abs/2009.01439 (visited on 05/22/2023), preprint.

[12] S. Andrews, K. Erleben, and Z. Ferguson, "Contact and friction simulation for computer graphics", in *ACM SIGGRAPH 2022 Courses*, Vancouver British Columbia Canada: ACM, Aug. 2, 2022, pp. 1–172, ISBN: 978-1-4503-9362-1. DOI: 10.1145/3532720.3535640. [Online]. Available: https://dl.acm.org/doi/10.1145/3532720.3535640 (visited on 04/28/2023).