OSCAR: Data-Driven Operational Space Control for Adaptive and Robust Robot Manipulation

Josiah Wong[1,2], Viktor Makoviychuk[2], Anima Anandkumar[2,3], Yuke Zhu[2,4]

***Abstract*— Learning performant robot manipulation policies can be challenging due to high-dimensional continuous actions and complex physics-based dynamics. This can be alleviated through intelligent choice of action space. Operational Space Control (OSC) has been used as an effective task-space controller for manipulation. Nonetheless, its strength depends on the underlying modeling fidelity, and is prone to failure when there are modeling errors. In this work, we propose OSC for Adaptation and Robustness (OSCAR), a data-driven variant of OSC that compensates for modeling errors by inferring relevant dynamics parameters from online trajectories. OSCAR decomposes dynamics learning into task-agnostic and task-specific phases, decoupling the dynamics dependencies of the robot and the extrinsics due to its environment. This structure enables robust zero-shot performance under out-of-distribution and rapid adaptation to significant domain shifts through additional finetuning. We evaluate our method on a variety of simulated manipulation problems, and find substantial improvements over an array of controller baselines. For more results and information, please visit `https://cremebrule.github.io/oscar/`.**

I. Introduction

Robust robot manipulation for real-world tasks is challenging as it requires controlling robots with many degrees of freedom to perform contact-rich interactions that can quickly adapt to varying conditions. While general reinforcement learning algorithms [1–3] can be employed for designing robot controllers from experiences, a crucial and often neglected design choice is the action space for specifying the desired motions of robots [4]. Recent work has shown promise in using abstract action representations, rather than low-level torque actuation, for expediting manipulation learning. A broad range of action representations have been examined, including task-space commands [4–9], latent-space action embeddings [10–13], and high-level goal specifications [14, 15]. These action abstractions have been shown to ease the exploration burdens of reinforcement learning and improve its sample efficiency.

Among these action representations, Operational Space Control (OSC) [16] has emerged as an effective task-space controller for contact-rich manipulation tasks [4, 6]. OSC parameterizes motor commands by end-effector displacement and maps these commands into deployable joint torques. It has many advantages with its dynamically consistent formalism, including modeling compliance, compressing the higher-dimensional non-linear N-DOF torque control into orthogonal 6-DOF actions, and accelerating learning by enabling agents to reason directly in the task space.

[1]Stanford University [2]NVIDIA [3]California Institute of Technology [4]The University of Texas at Austin. Correspondance to `jdwong@stanford.edu`

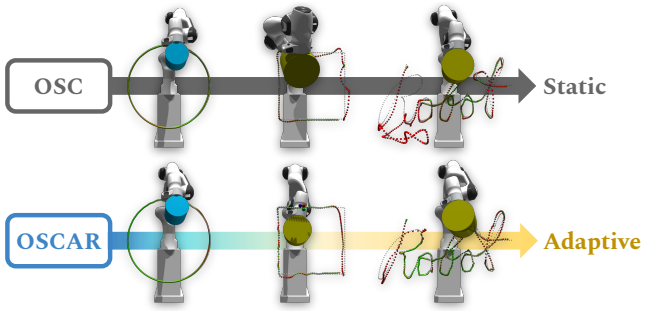


Fig. 1: **Adapting to Changing Dynamics.** Path Tracing is an example of a task requiring accurate trajectory motion and can be sensitive to dynamics parameters. For example, while in-distribution performance may be similar across multiple models (left), changing the train distribution end-effector weight (*blue*) and trajectories (*circles*) to unseen values (*yellow*, *squares*) can degrade performance if modeling errors are not accounted for. In contrast, OSCAR directly learns a dynamics representation online from scratch, enabling it to more robustly perform under zero-shot transfer conditions (middle) and quickly finetune under more extreme domain shifts, such as handwritten cursive (right).

However, OSC's benefits are not always realized. As a model-based controller, its practical strength heavily relies on a high-fidelity model of the *mass matrix*. This time-varying quantity accounts for the robot's mass distribution at its current configuration and is crucial for calculating torques. In the presence of inaccuracies in dynamics modeling, OSC's performance quickly deteriorates [17]. We illustrate this problem in a simple path tracing task in Fig. 1, where unmodeled extrinsics parameters such as friction and external forces significantly reduce the fidelity of the analytical mass matrix and deteriorate the tracking accuracy. The problem with modeling errors is exacerbated by the fact that controller design is often decoupled from policy learning, and becomes especially pronounced during task transfer settings such as simulation-to-real where there can be significant domain shifts. While a policy can finetune itself, mass matrix quality cannot be improved with data due to its analytical formulation.

How can we overcome the key limitation of OSC? We observe that its key element, the mass matrix, is subject to physics-based constraints as expressed by robot dynamics differential equations [18] in classical mechanics. Recent advances in physics-informed machine learning has developed a new family of neural networks [19–21] that learn such differential equations from data. In particular, Lutter et al. [22, 23] introduced Deep Lagrangian Networks (DeLAN) that directly infers the mass matrix from sampled trajectories. However, despite promising results on real robots, DeLAN is

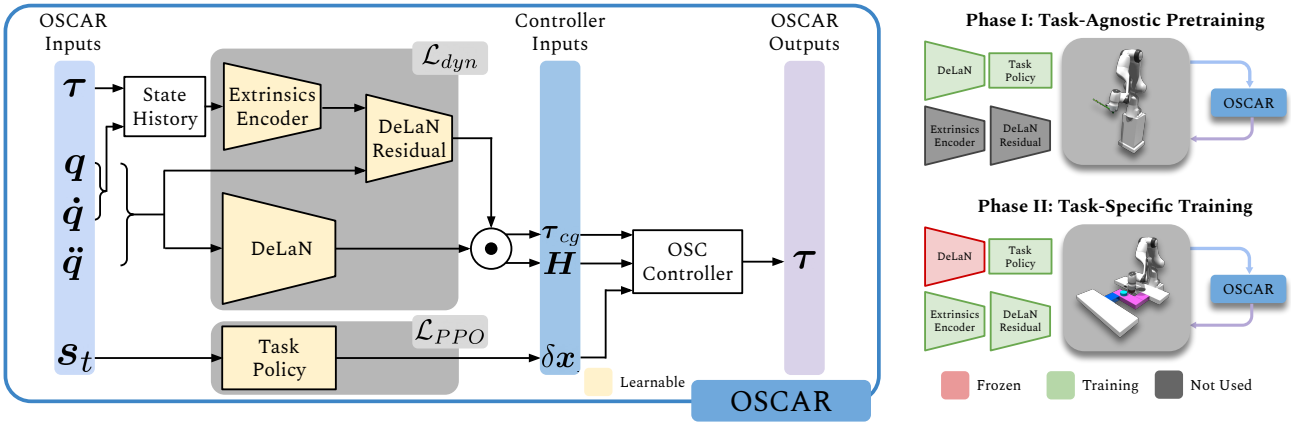


Fig. 2: **OSCAR Architecture.** Training OSCAR is split sequentially into a task-agnostic (top right) and task-specific (bottom right) phases. During the task-agnostic phase, an initial dynamics model is bootstrapped using DELAN using trajectories outputted by a policy being simultaneously trained to follow random waypoint trajectories. During the task-specific phase, we discard the original policy and freeze the base DELAN model, and then finetune the dynamics model using a residual while simultaneously learning a new task-relevant policy. The dynamics model's objective is dyanmics loss $\mathcal{L}_{dyn}$ whereas the policy is trained via RL using PPO loss $\mathcal{L}_{PPO}$.

limited by its modeling capacity and underlying assumptions, viz., that robots move in free space with constant mass and no external disturbances. This is unrealistic for most robot manipulation tasks, which require contact interaction with the environment. Hence, we need a more versatile and robust model for contact-rich manipulation tasks.

Our Approach: To this end, we introduce OSC for Adaptation and Robustness (OSCAR), a data-driven variant of OSC that leverages a neural network-based physics model to infer relevant modeling parameters and enable online adaptation to changing dynamics. OSCAR addresses DELAN's limitations by extending its formulation to be amenable to general dynamic settings such as robot manipulation.

Concretely, we introduce latent extrinsics inputs that capture task-specific environment factors and robot parameters, and design an encoder to infer these extrinsics from the state-action history. These extrinsics allow our model to infer wide variations in dynamics during training, and robustly work in out-of-distribution settings. Furthermore, we augment the dynamics model with a residual component. This design factorizes the dynamics model into a canonical task-agnostic component learned online from scratch using free-space motion and a constrained task-specific residual, allowing for fast adaptation to dynamics change through residual learning.

We evaluate OSCAR in three diverse manipulation tasks: Path Tracing, Cup Pouring, and Puck Pushing, all of which become especially challenging due to varying degrees of dynamics variations. We evaluate extensive baselines, and find that OSCAR is much more **stable**, being the only model to achieve task success on all tasks when evaluated on training distributions, **robust**, exhibiting significantly less policy degradation when evaluated in zero-shot under out-of-distribution, and **adaptive**, being the only model to reconverge to similar levels of task performance across all tasks when quickly finetuned under significant domain shifts.

II. BACKGROUND

A. Preliminaries

We model robot manipulation tasks as an infinite-horizon discrete-time Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \rho_0 \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(s_{t+1}|s_t, a_t)$ is the state transition probability distribution, $\mathcal{R}(s_t, a_t, s_{t+1})$ is the reward function, $\gamma \in [0,1)$ is the reward discount factor, and $\rho_0(\cdot)$ is the initial state distribution. Per timestep t, an agent observes s_t, deploys policy π to choose an action $a_t \sim \pi(a_t|s_t)$, and executes the action in the environment, observing the next state $s_{t+1} \sim \mathcal{T}(\cdot)$ and receiving reward $r_t = \mathcal{R}(\cdot)$. We seek to learn policy π that maximizes the discounted expected return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})]$.

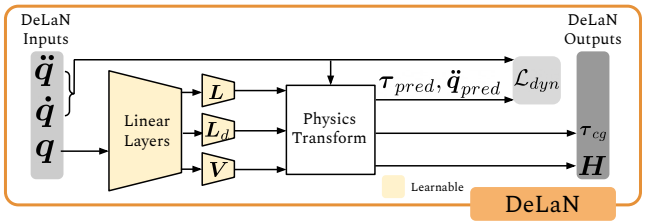
B. Operational Space Control (OSC)

OSC is a dynamically consistent controller that models compliant task-space motion. It is particularly useful for robot manipulation, where the robot's end-effector is often the critical task point and must produce compliant behavior for either safety or task-specific reasons. The control law is:

$$\boldsymbol{\tau} = \boldsymbol{J}_{ee}^{\top}(\boldsymbol{q})[\boldsymbol{\Lambda}_{ee}(\boldsymbol{q})[\boldsymbol{k}_p(\boldsymbol{x}_d - \boldsymbol{x}) - \boldsymbol{k}_v((\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}))]] \quad (1)$$

where the inertial matrix $\boldsymbol{\Lambda}_{ee} \in \mathbb{R}^{6\times6}$ and the Jacobian $\boldsymbol{J}_{ee}$ both specified in the end-effector frame maps the desired PD $\boldsymbol{k}_p, \boldsymbol{k}_v \in \mathbb{R}^6$ control of desired end-effector pose and velocity $\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d \in \mathbb{R}^6$ to joint-space control torques $\boldsymbol{\tau} \in \mathbb{R}^N$, where N is the number of robot joints. $\boldsymbol{k}_p$ and $\boldsymbol{k}_v$ specify the relative compliance of the controller [16]. These values can be either defined *a priori* or learned as part of the action space (VICES [4]). For VICES, the action space is $\mathbb{R}^{18}$, instead of the usual $\mathbb{R}^6$ with $\dot{\boldsymbol{x}}_d = 0$. This choice of action space has been shown to be especially advantageous for contact-rich manipulation [24]. For this reason, we apply VICES to all OSC-based models unless otherwise noted.

Crucially, $\boldsymbol{\Lambda}_{ee}$ depends on the *mass matrix* $\boldsymbol{H}(\boldsymbol{q})$, which in turn depends on the joint state $\boldsymbol{q}$ and potentially time if



$$\mathcal{L}_{dyn} = \mathcal{L}_{forward}(\ddot{\boldsymbol{q}}_{pred}, \ddot{\boldsymbol{q}}) + \mathcal{L}_{inverse}(\boldsymbol{\tau}_{pred}, \boldsymbol{\tau}) + \mathcal{L}_{energy}(\dot{\boldsymbol{E}}_{pred}, \dot{\boldsymbol{q}}^\top \boldsymbol{\tau})$$

Fig. 3: **DeLaN Architecture.** DeLaN takes robot joint states $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ as inputs, and outputs coriolis-gravity compensation torques $\boldsymbol{\tau}_{cg}$ and mass matrix $\boldsymbol{H}$. A shared linear layer core utilizes separate output heads to bootstrap different dynamics parameters: $\boldsymbol{H} = \boldsymbol{L} + \boldsymbol{L}^\top + \boldsymbol{L}_d$, and $\boldsymbol{g} = \frac{\partial}{\partial \boldsymbol{q}} \boldsymbol{V}$. These values are manipulated using forward (Eq. 2) and inverse (Eq. 3) dynamics equations to generate predicted values used in the dynamics loss $\mathcal{L}_{dyn}$.

the overall system mass is time-varying. Inaccurate modeling of the mass matrix can result in unstable controller behavior. This problem is precisely the challenge that we seek to address through learning $\boldsymbol{H}$ online from sampled trajectories.

C. Deep Lagrangian Networks (DeLaN)

Lutter et al. [22, 23] have shown that arbitrary rigid body dynamics can be inferred from free space motion via the Lagrangian mechanics, captured by the following forward and inverse dynamics equations:

$$f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}, \boldsymbol{H}, \boldsymbol{g}) = \boldsymbol{H}^{-1} \left(\boldsymbol{\tau} - \dot{\boldsymbol{H}}\dot{\boldsymbol{q}} + \frac{1}{2} \left(\frac{\partial}{\partial \boldsymbol{q}} ((\dot{\boldsymbol{q}}^\top \boldsymbol{H} \dot{\boldsymbol{q}}) \right)^\top - \boldsymbol{g} \right) \tag{2}$$

$$f^{-1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{H}, \boldsymbol{g}) = \boldsymbol{H}\ddot{\boldsymbol{q}} + \frac{d}{dt}(\boldsymbol{H})\dot{\boldsymbol{q}} - \frac{1}{2} \left(\frac{\partial}{\partial \boldsymbol{q}} (\dot{\boldsymbol{q}}^\top \boldsymbol{H} \dot{\boldsymbol{q}}) \right)^\top + \boldsymbol{g} \tag{3}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ are the system's generalized coordinates and their first- and second-order time derivatives, respectively, and $\boldsymbol{g}$ is the time-varying sum of potential forces. DeLaN parameterizes unknown functions $\boldsymbol{H}, \boldsymbol{g}$ with neural networks.

Given known values $\boldsymbol{q}_t$, $\dot{\boldsymbol{q}}_t$, $\ddot{\boldsymbol{q}}_t$, DeLaN predicts current torques $\boldsymbol{\tau}_{pred}$ using Eq. 3, and additionally uses observed torques $\boldsymbol{\tau}$ to predict joint accelerations $\ddot{\boldsymbol{q}}_{pred}$ using Eq. 2. DeLaN is trained with L2 regression loss with respect to known observed quantities, and includes forward, inverse, and energy conservation dynamics losses (Fig 3).

In order to allow efficient off-the-shelf back-propagation, the loss computations require analytical first-order derivative computations of $\frac{\partial}{\partial \boldsymbol{q}} \boldsymbol{H}$. Since $\boldsymbol{H}$ is parameterized by a neural network with $\boldsymbol{q}$ as inputs, this can be achieved by exclusively using fully-connected layers ("Lagrangian Layers") which allow analytical first-order partial derivatives to be computed during the forward pass.

III. Our Method (OSCAR)

We now propose OSCAR, a data-driven variant of OSC capable of leveraging the strengths of its controller formulation while removing the modeling burden of the high-fidelity and pre-defined mass matrix. We achieve this by learning a dynamics model that directly infers the mass matrix from online trajectories. Crucially, our dynamics model builds upon DeLaN and concretely improves its modeling capacity by (a) introducing latent extrinsics inputs that capture task-specific environment factors and robot parameters, which can be inferred directly from the state-action history, and (b) augmenting the dynamics model with a residual component, enabling fast adaptation to dynamics changes through residual learning (Fig. 2). Together, OSCAR enables end-to-end joint learning of the dynamics model and the policy, and can adapt to out-of-distribution environment variations. We describe each of these key features below.

A. Capturing Latent Extrinsics

A key limitation of DeLaN resides in its modeling capacity, which is bottlenecked by its dependency exclusively on $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$. While these values may be sufficient to model dynamics for a robot moving through free space in steady state, it can be problematic for settings where these values are not constant, such as during extended external impedance like grasping or applying the same model across a fleet of robots with varying individual dynamical properties.

OSCAR addresses this problem by implicitly inferring relevant extrinsic parameters directly from a robot's state-action history via learned latent embeddings. This follows the intuition that states are jointly dependent on both actions and underlying dynamics. Thus, learning correlations between recent states and actions can be a viable proxy for inferring extrinsic variations. This hypothesis has been validated by recent work [25] that has shown a similar method robust enough for direct simulation-to-real transfer.

To this end, we add an additional module to our dynamics model, which takes in the robot's recent joint states $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and low-level torques $\boldsymbol{\tau}$, and generates low-dimensional latent embeddings $\boldsymbol{z}$ that are directly fed into the DeLaN residual (Sec. III-B). Because the base network is restrained to inferring relevant dynamics from $\boldsymbol{q}$, augmenting inputs with $\boldsymbol{z}$ can increase the fidelity of the learned dynamics by allowing additional information flow from the state-action history. We use a shallow 4-layer MLP network as our encoder.

In order to leverage automatic differentiation for calculating the dynamics loss, we must analytically compute the first-order partial derivative $\frac{\partial}{\partial \boldsymbol{q}} \boldsymbol{z}$. However, we crucially assume that these latent embeddings capture environment extrinsics that are *independent* of $\boldsymbol{q}$, such that $\frac{\partial}{\partial \boldsymbol{q}} \boldsymbol{z} = \boldsymbol{0}$. This makes forward and gradient computations much simpler and also encourages the learned embeddings to be agnostic to spurious correlations with $\boldsymbol{q}$.

B. Residual-Augmented Mass Matrices

Under domain shifts that substantially change the system dynamics, it would be advantageous for a pretrained but possibly inaccurate dynamics model to quickly adapt in an online fashion. Therefore, OSCAR decouples dynamics learning into task-agnostic and task-specific components, inferring an initial reference estimate of the dynamics that can be applied to many models and finetuned on task-specific dynamics. We achieve this by decomposing the learned mass

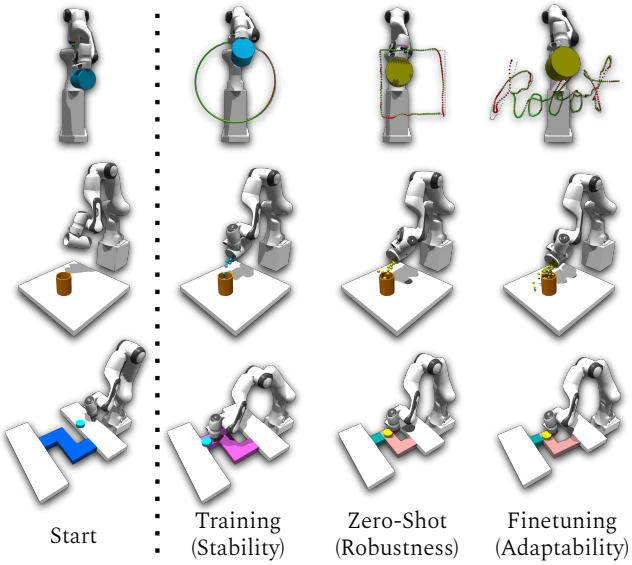


Fig. 4: **Tasks.** We present 3 manipulation tasks requiring both dexterity and precision: Path Tracing (top), Cup Pouring (middle), and Puck Pushing (bottom). Dynamics parameters such as mass and friction are randomized, and these tasks require compliance in order to successfully adapt to each instance. We evaluate three variants of each task: the training distribution to evaluate model stability (left), the out-of-distribution configuration with previously unseen dynamics to evaluate zero-shot robustness (middle), and significant out-of-distribution configurations to evaluate adaptability through finetuning (right). We symbolize some relevant parameter changes applied with distinguishing colors and visual changes.

matrix model $\boldsymbol{H}$ into a *task-agnostic* base $\hat{\boldsymbol{H}}$ and a small *task-specific* residual $\tilde{\boldsymbol{H}}$. Learning the task-agnostic base function mitigates the original modeling burden of OSC, and learning a task-specific residual can reduce the domain shift problem by adapting to the current dynamics.

We formulate our residual component $\tilde{\boldsymbol{H}}$ as a multiplicative residual outputting constrained scaling factors such that $\boldsymbol{H}$ is the element-wise multiplication of $\hat{\boldsymbol{H}}$ and $\tilde{\boldsymbol{H}}$:

$$\boldsymbol{H} = \hat{\boldsymbol{H}} \odot \tilde{\boldsymbol{H}}(\cdot;\phi), \quad ||\tilde{\boldsymbol{H}} - 1||_\infty < \varepsilon, \quad |\varepsilon| \text{ small} \tag{4}$$

where $\tilde{\boldsymbol{H}}$ is parameterized by a multi-layered neural network similar to DELAN, ϕ are the learned network weights, and $\odot$ is the element-wise multiplication operator. We choose this multiplicative structure instead of an additive one because we observe that the individual mass matrix elements span multiple orders of magnitude, thereby increasing the optimization difficulty for additive residuals whose outputs must similarly span such a large range of values.

We provide joint states $\boldsymbol{q}$, base estimate $\hat{\boldsymbol{H}}$, and latent extrinsics $\boldsymbol{z}$ (Sec. III-A) as the residual's inputs. $\boldsymbol{z}$ is included because we observe that environment extrinsics are often unique to the task at hand, and do not necessarily generalize across domains. For this same reason, we choose to exclude the latent extrinsics dependency during the $\hat{\boldsymbol{H}}$ pretraining process, which is meant to capture task-agnostic information about the mass matrix.

To enforce the residual's bounded influence, the residual neural network's output is passed sequentially through a scaled Tanh and Exponential layers. In this way, we can achieve a residual that is centered at 1 (no change to the base model) with symmetric log scale bounds.

A final requirement is to ensure that both the positive definiteness of $\boldsymbol{H}$ is preserved, and that $\boldsymbol{H}$'s first-order partial derivative $\frac{\partial}{\partial \boldsymbol{q}}$ can be analytically computed so that off-the-shelf auto-differentiation packages can still be used. The former is achieved by using a similar decomposition as $\hat{\boldsymbol{H}}$ and setting $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{L}} + \tilde{\boldsymbol{L}}^\top + \tilde{\boldsymbol{L}}_d$, where $\tilde{\boldsymbol{L}}$ is a lower diagonal matrix with zeros along the diagonal and $\tilde{\boldsymbol{L}}_d$ is a sparse matrix with only non-zero elements along its diagonal. For the latter, $\frac{\partial}{\partial \boldsymbol{q}}\boldsymbol{H}$ can be computed from Equation (4) using chain rule:

$$\frac{\partial}{\partial \boldsymbol{q}}\boldsymbol{H} = \frac{\partial}{\partial \boldsymbol{q}}\hat{\boldsymbol{H}} \odot \tilde{\boldsymbol{H}} + \hat{\boldsymbol{H}} \odot \left(\frac{\partial}{\partial \boldsymbol{q}}\tilde{\boldsymbol{L}} + \frac{\partial}{\partial \boldsymbol{q}}\tilde{\boldsymbol{L}}^\top + \frac{\partial}{\partial \boldsymbol{q}}\tilde{\boldsymbol{L}}_d \right) \tag{5}$$

By defining $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{L}}_d$ as Lagrangian Layers [22], we can tractably compute each of their first-order partial derivatives $\frac{\partial}{\partial \boldsymbol{q}}(\cdot)$ directly during the forward pass. We train our model using the same dynamics loss as DELAN (Fig. 3).

C. *Training* OSCAR

We seek to decompose our model into a task-agnostic core for maximizing modeling efficiency and a task-specific residual for improved per-task fidelity. We achieve this by splitting learning into pretrain and train phases:

Task-Agnostic Pretraining: We first learn an initial estimate of the mass matrix $\hat{\boldsymbol{H}}$ from scratch by training the core DELAN model in conjunction with a policy trained follow randomly generated straight line trajectories in free space. As this is intended to be a reference network, we do not utilize environment randomization nor end-effector masses in order to decouple these influences from initial dynamics learning.

Task-Specific Training: After pretraining, we discard the original policy and freeze the core DELAN backbone. For a new task, we proceed to train our residual extrinsics-aware model in conjunction with a new policy to learn residual mass matrix $\tilde{\boldsymbol{H}}$ to better capture the task-specific dynamics.

Our dynamics model runs at the same policy frequency (20Hz), and uses residual scaling limit of $\varepsilon = 0.1$.

IV. EXPERIMENTS

A. *Experimental Setup*

Tasks. We propose a set of simulated manipulation tasks that highlight the benefits of robust compliant control (Fig. 4). All experiments are conducted using the Franka Panda robot. We implement our environments using Isaac Gym [26], a high-fidelity simulator that has been shown to enable simulation-to-real transfer learning to physical hardware [27, 28].

Path Tracing: The robot must follow a set of procedurally generated way-points defining a parameterized path. Sampled paths during training include circles of varying sizes with randomized positions and orientations. The robot also has a sealed container attached to its end-effector containing an unknown mass of randomized weight. Task observations include agent end-effector and goal poses. This task focuses on the robot's ability to accurately follow desired trajectories while being robust to varying impedance.

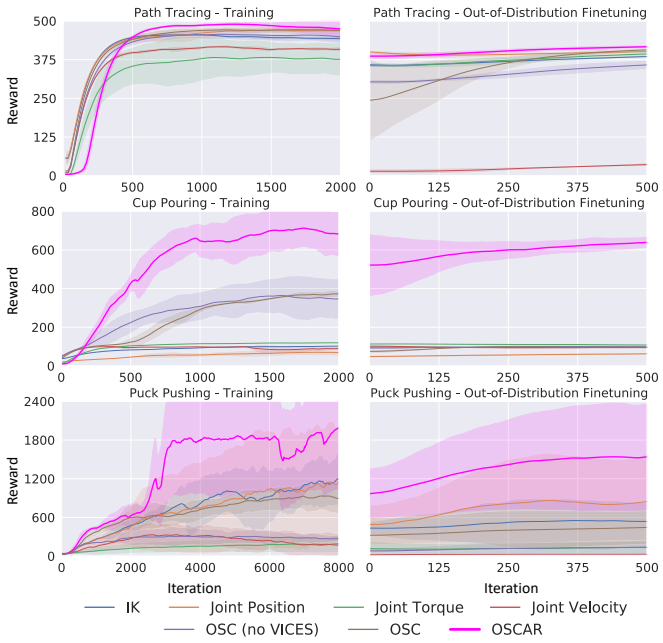


Fig. 5: **Task Results.** (Left) When initially trained on each task, OSCAR already outperforms the strongest baselines without leveraging any privileged model information such as mass matrix $\boldsymbol{H}$ used by analytical OSC baselines during training. (Right) When finetuned under significant domain shifts, OSCAR re-converges more quickly to more performant levels compared to all the baselines.

TABLE I: **Zero-Shot Generalization Results**

Model	Path Tracing MSE (mm)	Cup Pouring % Filled	Puck Pushing % Completed
Joint Position	24.4 ± 34.2	1.9 ± 2.7	22.8 ± 16.1
Joint Velocity	482.2 ± 175.4	0.0 ± 0.0	0.9 ± 0.5
Joint Torque	37.5 ± 10.8	0.0 ± 0.0	8.4 ± 5.9
IK	27.0 ± 6.1	0.0 ± 0.0	33.4 ± 1.1
OSC (no VICES)	44.6 ± 7.4	0.4 ± 0.4	7.7 ± 2.1
OSC	26.1 ± 5.3	0.6 ± 0.2	46.5 ± 5.3
OSCAR (Ours)	$\mathbf{20.0 \pm 2.3}$	$\mathbf{73.8 \pm 9.2}$	$\mathbf{76.0 \pm 14.2}$
Δ over the Best Baseline	**+18%**	**+3780%**	**+63%**

Cup Pouring: The robot holds a pitcher containing small particles and must carefully pour them into a cup located on the table while minimizing the amount of spillage. The cup's diameter, height, and position are randomized between episodes. Task observations include end-effector pose, pitcher tilt, cup position, cup radius, cup height, and the proportion of particles both in and outside of the cup. This task focuses on the robot's ability to respond to changing dynamics.

Puck Pushing: The robot must carefully push a puck between tables while avoiding knocking the puck off the platform. Task observations include end-effector pose, puck position and tilt, relative puck position to next path waypoint, and the proportion of the path completed. This task focuses on the robot's ability to adapt to sustained external contact.

In all of the tasks, the robot's initial pose, joint friction, damping, armature, and minimum inertia are randomized between episodes. Specific parameters and reward functions can be found on our website.

Baselines. We compare OSCAR against several baseline controllers, including joint-space options Joint Position, Joint Velocity, Joint Torque and task-space options Inverse Kinematics (IK), OSC (no VICES), OSC. Among these controllers, the joint-space controllers must operate in the highly nonlinear joint actuation, and IK cannot model compliance.

The torque-based controller baselines are automatically provided with rough estimations of gravitational compensation torques $\tau_{cg} = \sum_{i=1}^{N} \boldsymbol{J}_i{}^{\top} m_i$, where $\boldsymbol{J}_i$ is the Jacobian and m_i is the mass for the i-th robot link. For OSCAR, τ_{cg} is generated directly from the learned dynamics model. In contrast, the IK, Joint Velocity, and Joint Position controllers do not operate in torque space, and so we disable gravity for these controllers. They have a distinct advantage over the torque-based controllers because their learned policies do not have to account for the dynamics influences from gravity. OSC baselines are provided the analytical mass matrix. whereas OSCAR does not have access to any analytical dynamics parameters from the robot model.

B. Performance: Train-Distribution Evaluation

We first consider the normal training performance of each controller on each task. We train our policies using PPO [29] with 2048 parallelized environments using 3 random seeds, and utilize identical hyperparameters across all models for fair comparison (Fig. 5). We find that OSC is often the strongest baseline across all tasks and consistently outperforms OSC without VICES. This result validates OSC's strengths as a task-space compliant controller and highlights the advantages of dynamically choosing compliance gains to enable policy adaptation to changing dynamics.

OSCAR outperforms all baselines across all tasks, with significant improvements in the more complex Cup Pouring (over **90**%) and Puck Pushing tasks (over **60**%) compared to the next best baseline. This result indicates that policy learning is not hindered by the OSCAR's learned dynamics model, and in fact benefits from the learned task-specific dynamics. We also note that while the decomposition of $\boldsymbol{\tau}_{cg}$ is not supervised during training, the learned values are sufficiently stable for OSCAR to succeed at each task and still significantly outperform other baselines.

C. Robustness: Out-of-Distribution Zero-Shot Evaluation

After training, we examine each trained model's robustness to novel task instances in a zero-shot setting. We modify each task's parameters to out-of-distribution combinations previously unseen during training. We maximize the influence of these extrinsics by setting dynamics parameters to their maximum training range values, and significantly increasing relevant masses over the training maximum value. We measure relevant success metrics for each task, and find that OSCAR is the most robust to policy degradation, outperforming the next best baseline by **63**% on the Puck Pushing task and **3780**% on the Cup Pouring task (Table I). These results show that OSCAR can be readily deployed in previously unseen dynamics while maintaining high task performance.

D. Adaptability: Out-of-Distribution Finetuning

Finally, we evaluate each model's ability to rapidly adapt to distribution shifts. In this experiment, we aggressively

TABLE II: **Ablation Study on Cup Pouring Task**

Variant	Train Return	Zero-shot Degradation
OSCAR (Ours)	844 ± 19	$\mathbf{-62}$
Additive residual	813 ± 24	-109
No extrinsics	$\mathbf{845 \pm 45}$	-171
No residual + finetune pretrain	711 ± 76	-166
No residual + freeze pretrain	774 ± 99	-105
No residual + no pretrain	760 ± 20	-95
Δ over the Best Baseline	-0.1%	$+\mathbf{34.7}\%$

modify the task parameters and then allow each trained model to finetune under the new settings. To reflect more realistic transfer settings with limited resource bandwidth, we train using only 4 parallel environments instead of 2048 as before. We find that OSCAR re-converges much more quickly compared to all other baselines and also achieves over **500**% for Cup Pouring and **90**% for Puck Pushing performance improvements over the next best baseline (Fig. 5). These results suggest that OSCAR helps overcome the exploration burden for difficult task instances by leveraging a previously trained model and further adapting it online. This property is appealing for task transfer, where OSCAR can leverage the advantages of large-scale simulations while quickly adjusting to new environment conditions in a sample-efficient manner.

E. Ablation Study

Lastly, we compare OSCAR against multiple ablative self-baselines on the Cup Pouring task to validate some key design decisions. We report both the training distribution performance and zero-shot out-of-distribution degradation (Table II). We find that while our model is marginally outperformed by some other variant during training, it is the most robust under zero-shot domain shift and results in **34.7**% less degradation compared to the next strongest ablative model. These results validate the key modeling components of OSCAR: namely, the multiplicative residual (vs. additive), extrinsics encoder (vs. no extrinsics), and two-step task-agnostic task-specific method used (vs. using no residual and manipulating the base DeLAN structure instead).

V. Related Work

Action Space for Learning Robot Control. Choosing an appropriate action space for learning methods in robotics can be nontrivial. While joint-space commands can be immediately deployed on a robot, they are high-dimensional and nonlinear. Recent work has demonstrated that the abstraction provided by task-space controllers, such as inverse kinematics (IK) [30] and operational space control (OSC) [16], can improve policy performance and reduce sample complexity [4, 6, 24, 31]. Nonetheless, these approaches are heavily model-based and require parameter tuning to overcome kinematic redundancies and modeling errors [32, 33]. Despite OSC's more expressive formulation that models compliance, prior work has primarily focused on improving IK [34–37]. A few works exploring OSC have sought to learn its parameters in data-driven ways, such as applying reinforcement learning (RL) to track desired trajectories [38] or learning the impedance gains for adapting to task-specific dynamics [4]. Unlike these methods which are limited to gain tuning and require knowledge of the underlying dynamics, we seek to completely eliminate the modeling burdens and infer dynamics from trajectories.

Deep Learning for Dynamical Systems. In contrast to policy learning methods that seek to *adapt* to environment dynamics, a parallel line of work has explored directly modeling these system dynamics as learnable ordinary differential equations (ODEs) with deep neural networks. While the majority of these works seek to model arbitrary physical dynamics [22, 23, 39, 40], some have explored specific complex physical phenomena such as contact [41–43] and fluid dynamics [44, 45]. These works leverage strong physical priors to enforce model plausibility. While these works have demonstrated promising results in restrictive domains, they have yet to show success for realistic contact-rich dynamics, such as high degree-of-freedom robot arms physically interacting with objects. Our work builds upon Lutter et al. [22, 23]'s DeLAN model. We augment its modeling capacity to perform joint policy and dynamics learning end-to-end for contact-rich manipulation tasks.

System Identification in Robotics. Our work is also relevant to methods for system identification (sysID), which seek to estimate accurate models of dynamics from data. Classical methods on sysID [46–48] define a priori models and seek to estimate its unknown parameters from sampled data. More recent data-driven methods have relaxed the modeling burdens and proposed to train deep neural networks to capture relevant task dynamics parameters either directly [27, 49] or via latent representations [25, 50]. Recent work on simulation-to-real transfer has focused on estimating distributions of simulated dynamics parameters [51] or allowing end-to-end differentiation through simulation [52, 53] for tuning the simulation model's fidelity. In a similar vein as these works, we leverage simulation with varied dynamics to allow our learned dynamics model to infer both direct and latent representations of these parameters. However, in contrast to most of these works that infer relevant parameters in an end-to-end fashion, we decouple the process into task-agnostic and task-specific phases, allowing our model to generate a reference representation that can be tuned to specific tasks.

VI. Conclusion

We showcased OSCAR, a data-driven method online learning of dynamics parameters for OSC from scratch, removing OSC's modeling burden and accelerating simultaneous end-to-end training of a task policy and a dynamics model. We showed that compared to extensive baselines, OSCAR shows better performance during training, less degradation in zero-shot out-of-distribution, and better adaptation when finetuned under significant domain shifts. While we were only able to validate OSCAR's sim2sim transfer due to limited hardware access during the pandemic, we hope to soon evaluate on sim2real transfer and further develop our method as a means to deploying data-driven OSC control to real robots.

ACKNOWLEDGMENT

We would like to thank Jim Fan and the NVIDIA AI-ALGO team for their insightful feedback and discussion, and the NVIDIA IsaacGym simulation team for providing technical support.

REFERENCES

[1] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, 2018. eprint: arXiv:1801.01290.

[2] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap, *Distributed distributional deterministic policy gradients*, 2018. eprint: arXiv:1804.08617.

[3] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, *Benchmarking reinforcement learning algorithms on real-world robots*, 2018. eprint: arXiv:1809.07731.

[4] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, *Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks*, 2019. eprint: arXiv:1906.08880.

[5] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation", in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4639–4644.

[6] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, *Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks*, 2018. eprint: arXiv:1810.10191.

[7] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Task-space control of robot manipulators with null-space compliance", *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 493–506, 2014.

[8] B. Xian, M. de Queiroz, D. Dawson, and I. Walker, "Task-space tracking control of robot manipulators via quaternion feedback", *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 160–167, 2004.

[9] S. S. Ge, C. Hang, and L. Woon, "Adaptive neural network control of robot manipulators in task space", *IEEE Transactions on Industrial Electronics*, vol. 44, no. 6, pp. 746–752, 1997.

[10] A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, and A. Garg, *Laser: Learning a latent action space for efficient reinforcement learning*, 2021. eprint: arXiv:2103.15793.

[11] M. Li, D. P. Losey, J. Bohg, and D. Sadigh, *Learning user-preferred mappings for intuitive robot control*, 2020. eprint: arXiv:2007.11627.

[12] S. Karamcheti, A. J. Zhai, D. P. Losey, and D. Sadigh, *Learning visually guided latent actions for assistive teleoperation*, 2021. eprint: arXiv:2105.00580.

[13] R. Pahič, Z. Lončarević, A. Gams, and A. Ude, "Robot skill learning in latent space of a deep autoencoder neural network", *Robotics and Autonomous Systems*, vol. 135, p. 103 690, 2021.

[14] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, and D. Kragic, "Latent space roadmap for visual action planning of deformable and rigid object manipulation", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5619–5626.

[15] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar, *Learning robotic manipulation through visual planning and acting*, 2019. eprint: arXiv:1905.04411.

[16] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation", *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[17] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison", *The International Journal of Robotics Research*, vol. 27, p. 737, Jun. 2008.

[18] S. B. Niku, *Introduction to robotics: analysis, systems, applications*. Prentice hall New Jersey, 2001, vol. 7.

[19] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, *Neural ordinary differential equations*, 2018. eprint: arXiv:1806.07366.

[20] Y. Shao, M. Hellström, P. D. Mitev, L. Knijff, and C. Zhang, "Pinn: A python library for building atomic neural networks of molecules and materials", 2019. eprint: arXiv:1910.03376.

[21] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, *Fourier neural operator for parametric partial differential equations*, 2020. eprint: arXiv:2010.08895.

[22] M. Lutter, C. Ritter, and J. Peters, *Deep lagrangian networks: Using physics as model prior for deep learning*, 2019. eprint: arXiv:1907.04490.

[23] M. Lutter, K. Listmann, and J. Peters, *Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems*, 2019. eprint: arXiv:1907.04489.

[24] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, *Reinforcement learning on variable impedance controller for high-precision robotic assembly*, 2019. eprint: arXiv:1903.01066.

[25] A. Kumar, Z. Fu, D. Pathak, and J. Malik, *Rma: Rapid motor adaptation for legged robots*, 2021. eprint: arXiv:2107.04034.

[26] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, *Isaac gym: High performance gpu-based physics simulation for robot learning*, 2021. eprint: arXiv:2108.10470.

[27] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, *Closing the sim-to-real loop: Adapting simulation randomization with real world experience*, 2018. eprint: arXiv:1810.05687.

[28] Y. Narang, B. Sundaralingam, M. Macklin, A. Mousavian, and D. Fox, *Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections*, 2021. eprint: arXiv:2103.16747.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. eprint: arXiv:1707.06347.

[30] A. A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots", *Robotics and Automation, IEEE Journal of*, vol. 1, pp. 14–20, Apr. 1985.

[31] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, *Robosuite: A modular simulation framework and benchmark for robot learning*, 2020. eprint: arXiv:2009.12293.

[32] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics", vol. 1, Feb. 2001, 298–303 vol.1.

[33] J. Peters and S. Schaal, "Learning to control in operational space", *The International Journal of Robotics Research*, vol. 27, p. 197, Feb. 2008.

[34] P. Srisuk, A. Sento, and Y. Kitjaidure, "Inverse kinematics solution using neural networks from forward kinematics equations", in *2017 9th International Conference on Knowledge and Smart Technology (KST)*, 2017, pp. 61–65.

[35] B. Karlik and S. Aydin, "An improved approach to the solution of inverse kinematics problems for robot manipulators", *Engineering Applications of Artificial Intelligence*, vol. 13, no. 2, pp. 159–164, 2000.

[36] Q. Chen, S. Zhu, and X. Zhang, "Improved inverse kinematics algorithm using screw theory for a six-dof robot manipulator", *International Journal of Advanced Robotic Systems*, vol. 12, p. 1, Oct. 2015.

[37] R. Köker, "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization", *Information Sciences*, vol. 222, pp. 528–543, 2013, Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.

[38] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control", *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007), 745-750 (2007)*, Jan. 2007.

[39] M. Finzi, K. A. Wang, and A. G. Wilson, *Simplifying hamiltonian and lagrangian neural networks via explicit constraints*, 2020. eprint: arXiv:2010.13581.

[40] Y. D. Zhong, B. Dey, and A. Chakraborty, *A differentiable contact model to extend lagrangian and hamiltonian neural networks for modeling hybrid dynamics*, 2021. eprint: arXiv:2102.06794.

[41] A. Hochlehnert, A. Terenin, S. Saemundsson, and M. Deisenroth, "Learning contact dynamics using physically structured neural networks", in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, A. Banerjee and K. Fukumizu, Eds., ser. Proceedings of Machine Learning Research, vol. 130, PMLR, 13–15 Apr 2021, pp. 2152–2160.

[42] S. Pfrommer, M. Halm, and M. Posa, *Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations*, 2020. eprint: arXiv:2009.11193.

[43] M. Parmar, M. Halm, and M. Posa, *Fundamental challenges in deep learning for stiff contact dynamics*, 2021. eprint: arXiv:2103.15406.

[44] H. Wessels, C. Weißenfels, and P. Wriggers, "The neural particle method – an updated lagrangian physics informed neural network for computational fluid dynamics", *Computer Methods in Applied Mechanics and Engineering*, vol. 368, p. 113 127, 2020.

[45] G. D. Portwood, P. P. Mitra, M. D. Ribeiro, T. M. Nguyen, B. T. Nadiga, J. A. Saenz, M. Chertkov, A. Garg, A. Anandkumar, A. Dengel, R. Baraniuk, and D. P. Schmidt, *Turbulence forecasting via neural ode*, 2019. eprint: arXiv:1911.05180.

[46] J. L. Melsa, *System identification*. Academic Press, 1971.

[47] K. Åström and P. Eykhoff, "System identification—a survey", *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[48] L. Ljung, "System identification", in *Signal analysis and prediction*, Springer, 1998, pp. 163–173.

[49] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, *Sim-to-real transfer for biped locomotion*, 2019. eprint: arXiv:1903.01390.

[50] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals", 2014. eprint: arXiv:1407.3501.

[51] F. Ramos, R. C. Possas, and D. Fox, "Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators", 2019. eprint: arXiv:1906.01728.

[52] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, *Brax - a differentiable physics engine for large scale rigid body simulation*, version 0.0.4, 2021.

[53] E. Heiden, M. Macklin, Y. Narang, D. Fox, A. Garg, and F. Ramos, *Disect: A differentiable simulation engine for autonomous robotic cutting*, 2021. eprint: arXiv:2105.12244.