2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Fast-Grasp'D: Dexterous Multi-finger Grasp Generation Through Differentiable Simulation

Dylan Turpin[1,2,3], Tao Zhong[1,2], Shutong Zhang[1,2], Guanglei Zhu[1,2], Eric Heiden[3], Miles Macklin[3], Stavros Tsogkas[1,4*], Sven Dickinson[1,2,4], Animesh Garg[1,2,3]

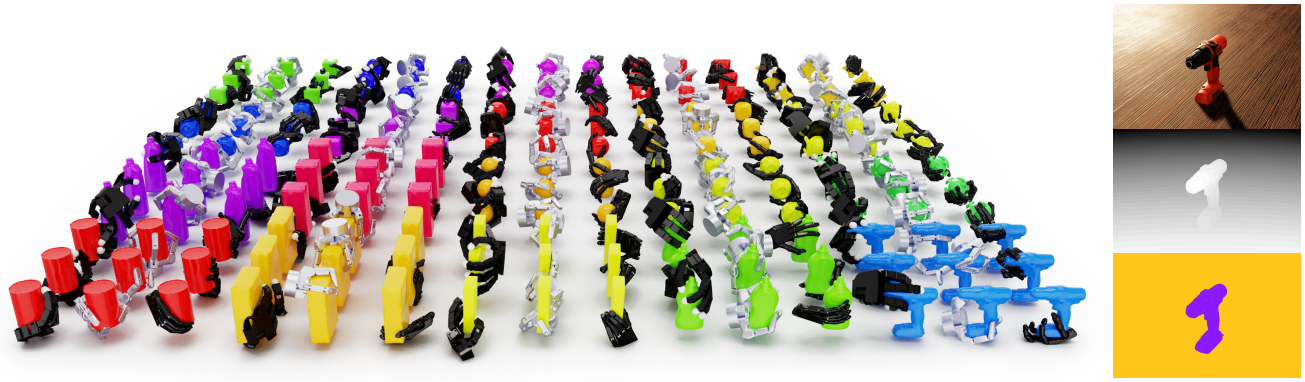

Fig. 1: The Grasp'D-1M dataset contains one million unique grasps, each with multi-modal visual inputs for training vision-based robotic grasping. We synthesize these grasps with a new differentiable grasping simulator, *Fast*-Grasp'D. Gradient information accelerates the grasp search, allowing us to search the full-DOF space (without eigengrasps) and simulate thousands of contacts to produce a dataset of contact-rich, stable grasps that can improve any learned grasping pipeline.

***Abstract*— Multi-finger grasping relies on high quality training data, which is hard to obtain: human data is hard to transfer and synthetic data relies on simplifying assumptions that reduce grasp quality. By making grasp simulation differentiable, and contact dynamics amenable to gradient-based optimization, we accelerate the search for high-quality grasps with fewer limiting assumptions. We present Grasp'D-1M: a large-scale dataset for multi-finger robotic grasping, synthesized with *Fast*-Grasp'D, a novel differentiable grasping simulator. Grasp'D-1M contains one million training examples for three robotic hands (three, four and five-fingered), each with multimodal visual inputs (RGB+depth+segmentation, available in mono and stereo). Grasp synthesis with *Fast*-Grasp'D is 10x faster than GraspIt! [1] and 20x faster than the prior Grasp'D differentiable simulator [2]. Generated grasps are more stable and contact-rich than GraspIt! grasps, regardless of the distance threshold used for contact generation. We validate the usefulness of our dataset by retraining an existing vision-based grasping pipeline [3] on Grasp'D-1M, and showing a dramatic increase in model performance, predicting grasps with 30% more contact, a 33% higher epsilon metric, and 35% lower simulated displacement. Additional details at fast-graspd.github.io.**

I. Introduction

Multi-finger robotic grasping is necessary for the effective operation of robots in everyday environments, which are filled with objects and affordances built for human hands. Recent works [3–5] in vision-based robotic grasping learn mappings: (1) from visual input to gripper poses (*direct regression*); or (2) from visual input and a proposed gripper pose to a score (*sampling*). A high-quality, large-scale dataset is necessary to learn either one of these mappings.

Such data are typically generated synthetically. Datasets of real human grasps can be captured [6, 7], and research on the grasp transfer problem considers the best way to adapt a human grasp pose to a robot [8–10]. However, this remains an open problem, and may be impractical in cases where the difference between human and robot hand morphology is noticeable (see Fig. 8 of [10]). Real robot trials offer a way of evaluating proposed grasps, but are usually considered too slow to use inside of a sampling loop.

Black-box optimization (e.g., simulated annealing [1]) takes many samples to find good grasps in the high-dimensional pose space of multi-finger grippers. Simulation-based metrics – widely used for parallel-jaw grasping due to their greater physical fidelity – are too expensive to compute at each step. Instead, we still rely largely on analytic metrics, and even then, limit the search to a low-dimensional subspace and pre-specify a handful of possible contact locations. This results in poor quality grasps. It is unlikely that conformal grasps exist in any low-dimensional subspace we choose to search. Simple refinement methods (e.g., *autograsp* in [1] or the differentiable layer in [3]) create some contact by closing the fingers, but rarely discover high-contact grasps.

Our previous work [2] shared a similar motivation, but was impractically slow (5 minutes per grasp), did not include data for robotic hands, and did not evaluate whether better synthesis of training data translates to improved model performance. We address these limitations with algorithmic changes to our simulator. We use an integrator based on position-based dynamics, known to be stable and robust in contact-rich scenarios and allowing us to forgo a problem relaxation

[1]University of Toronto, [2] Vector Institute, [3]Nvidia, [4]Samsung.
Correspondence: {dylanturpin, garg}@cs.toronto.edu
Dickinson and Tsogkas contributed to this work in their capacity as Professor and Adjunct Professor, respectively, at the University of Toronto. The views expressed (or the conclusions reached) are their own and do not necessarily represent the views of Samsung Research America, Inc.

Dataset	Robotic Hands (# of Fingers)	Visual inputs	Available input modalities	Generation method	Number of unique grasps	Number of training examples
Multi-FinGAN [11]	Barrett (3)	✓	RGBD, segmentation	GraspIt! [1]	1,355	4,990
DDGC [3]	Barrett (3)	✓	RGBD, segmentation	GraspIt! [1]	6,793	185,598
Columbia Grasp Database [12]	Barrett (3)	x	(none)	GraspIt! [1]	158,006	(none)
Grasp'D-1M (ours)	**Barrett (3), Allegro (4) Shadow (5)**	✓	**RGBD, segmentation, 2D/3D bbox (in mono+stereo)**	Differentiable Simulation	**1,000,000**	**1,000,000**

TABLE I: Datasets of multi-finger robotic grasps for training vision-based grasping are uncommon, limited in size (especially when considering the number of unique grasps, which are reused with multiple scenes or camera angles) and contain grasps whose quality is limited by the assumptions necessary for sampling-based planning with GraspIt! [1] to succeed.

based on contact-invariant optimization [13] (thereby reducing optimization variables from thousands to tens). We represent the object-to-be-grasped with a mesh (rather than a fixed sized grid), and compute smoothed Phong [14] signed distances on the fly. This leads to more accurate signed-distances and computation time that scales with mesh complexity. We summarize our contributions as follows:

1) We introduce the *Fast*-Grasp'D simulator and pipeline for differentiable grasp synthesis (10× faster than GraspIt! [1] and 20× faster than [2]) with a contact model well-suited to learning by gradient descent.
2) We introduce the Grasp'D-1M dataset of one million unique grasps with multi-modal visual input for vision-based multi-finger robotic grasping.
3) We perform a thorough evaluation of our synthesized grasps as compared to GraspIt!, showing our grasps provide more contact and higher stability, regardless of the distance threshold used for contact generation.
4) Finally, we demonstrate the value of Grasp'D-1M by using it to improve the performance of vision-based grasp prediction, inducing 30% more contact, a 33% higher epsilon metric and 35% lower simulated displacement.

II. Related Work

Vision-based grasp prediction – Modern approaches [5, 15] to grasp prediction learn a mapping from visual inputs to grasps (or to a quality function used alongside a sampler) by training on a dataset of positive examples. [3, 11, 16] employ GAN-style models to predict stable grasps from RGBD inputs. [17, 18] take an implicit approach to grasp representation by learning to jointly predict signed distances for a gripper and object to be grasped. Recent works on parallel-jaw grasping [19–22] use datasets derived from simulation [23, 24], which have better physical fidelity [20, 24, 25] and more intuitive plausibility [23] than analytic datasets. In contrast, multi-finger robotic grasp prediction continues to be trained on analytically synthesized datasets. [3, 11, 17, 18, 26–28] all use analytically synthesized datasets from the GraspIt! simulator [1]. We aim to improve vision-based multi-finger grasping by using simulation to generate better datasets (and using gradient-based optimization to make the higher computational cost of simulation affordable).

Multi-finger robotic grasping datasets – Only a handful of datasets exist for multi-finger grippers (see Table I). Of these, most include only the three-finger Barrett hand [3, 11, 12] or do not include visual inputs to learn from [8, 12, 29]. The Grasp'D-1M dataset contains grasps for grippers with three (Barrett), four (Allegro), and five (Shadow) fingers, along with a variety of multi-modal visual inputs to learn from. Furthermore, through differentiable simulation, we are able to synthesize more physically-plausible, stable, and contact-rich grasps than can be found with the analytic grasp synthesis [1, 30] used to generate other datasets.

Grasp synthesis. Since human grasps are difficult to transfer to robotic hands, and gathering real robot data is expensive (and presupposes a way to generate trial grasps), robotic grasping datasets often rely on artificial grasp synthesis. Analytic synthesis, which optimizes a handcrafted metric to find stable grasps, has been successfully applied to *parallel-jaw* grippers (based on grasp wrench space analysis [1, 12, 31], robust grasp wrench space analysis [32, 33], or caging [34, 35]). While they are more computationally costly, simulation-based metrics better align with human judgement [23, 36] and with real world performance [19, 20, 24, 25]. Unfortunately, this higher computational cost has delayed the adoption of simulation-based synthesis for multi-finger hands. Sample-intensive black-box optimization in a high-dimensional pose space renders simulated metrics too expensive. In fact, for high-DOF hands, optimizing over even simple analytic metrics is usually impractical without limiting search to a low-dimensional subspace [37] and considering only a handful of pre-specified contact locations. We introduce a differentiable simulation-based metric. Gradient-based optimization reduces the number of samples required, and GPU parallelism makes simulation fast enough that we can search the full grasp space and simulate thousands of contacts in order to find contact-rich, physically plausible grasps.

Differentiable Physics – While there has been brisk recent progress in differentiable physics engines [38–46], myriad limitations render them inadequate for our application. Brax [41] and the Tiny Differentiable Simulator [44] only support collision primitives and cannot model collisions between complex meshes. Nimblephysics [42] supports mesh-to-mesh collision, but cannot handle cases where the gradient of contact normals with respect to position is zero (e.g., on a mesh face). While its analytic computation of gradients is fast, Nimblephysics requires the manual derivation of the backward pass when new simulation functionality is added. Instead, in this work, similar to Brax [41] and Taichi [39], we rely on automatic differentiation to translate user-defined simulation code from high-level Python to kernel codes that run very efficiently on GPUs.

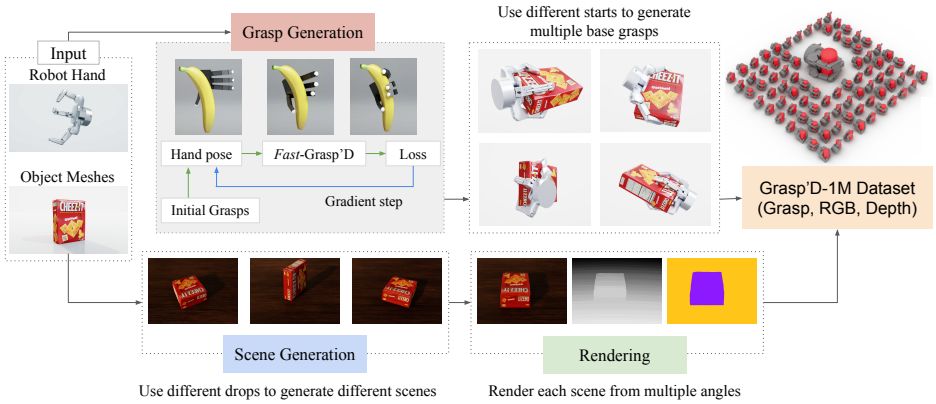

Fig. 2: Our grasp synthesis pipeline generates the Grasp'D-1M dataset of one million unique grasps in three stages. *(1) Grasp generation*: For any provided (robot hand, object) pair, we generate a set of base grasps by gradient descent over an objective computed by *Fast*-Grasp'D, our fast and differentiable grasping simulator. *(2) Scene generation*: We simulate multiple drops of each object onto a table to create scenes with different object poses and transfer base grasps to these scenes. *(3) Rendering*: Finally, we render each scene (RGB, depth, segmentation, 2D/3D bounding boxes in mono+stereo) from multiple camera angles.

Differentiable Grasping – Differentiable multi-finger grasp synthesis is a less explored domain. [47] and [48] formulate differentiable force closure metrics and use gradient-based optimization to synthesize grasps with the Shadow and MANO [49] hands, respectively. This allows analytic synthesis to search the full-dimensional pose space of a high-DOF hand, yet still exhibits the usual drawbacks of analytic metrics. [47] assumes that contact is limited to 45 manually labelled points, and [48] assumes zero friction, uniform force magnitude across all contacts, and only scales to grasps with a few point contacts (with three contacts it takes ~ 40 minutes to find 5 acceptable grasps). Our previous work [2] formulated a differentiable simulation-based metric able to scale to thousands of contacts to approximate surface contact and generate plausible, contact-rich grasps. However, Grasp'D [2] was still slow (~ 5 minutes per grasp), focused on human rather than robotic hands, did not release a dataset or quantitative evaluation for robot grippers, and did not evaluate the benefit of actually training vision-based pipelines on the generated data. *Fast*-Grasp'D addresses these concerns with up to a $30\times$ speedup ($\sim$10s per grasp online or $\sim$1s amortized), a large-scale dataset for vision-based robotic grasping, and a thorough evaluation of an existing vision-based grasping pipeline retrained using *Fast*-Grasp'D data. Two concurrent works [50, 51] also propose differentiable stability metrics, but their fidelity is limited by considering only a handful of contact locations at a time. We include a favourable comparison to [50] in Section V. [51] has not yet released data.

III. *Fast*-Grasp'D: Differentiable Grasp Synthesis

We present a method for grasp synthesis using an input object mesh and hand model (represented by a mesh and an articulation chain with meshes at the links), and generate a physically-plausible stable grasp as a base pose and joint angles of the hand. This is achieved by gradient descent over a stability metric computed by our differentiable grasping simulator, *Fast*-Grasp'D. The final grasp is dependent on the pose initialization of the hand, so different grasps can be recovered by sampling different starting poses. We extend differentiable grasp synthesis from previous work [2] with algorithmic changes to achieve a $20\times$ speedup in performance, generating a grasp in about one second. Namely, we improve the method's integration scheme (to ensure stable optimization without introducing additional relaxation variables) and object representation (to scale computation with object complexity). We also adapt the concepts of signed-distance function dilation and *leaky gradient* to position-based dynamics. We build on the Warp [52] framework, which supports fast auto-differentiation and GPU acceleration.

A. *From Grasp'D to Fast-Grasp'D*

We outline the challenges that motivate our design. ***Gradient-friendly contact dynamics for mesh inputs*** – The algorithms usually employed for mesh-to-mesh collision rely on operations (e.g., tree-traversal) that are hard to differentiate through. A formulation of contact constraints based on signed distance functions (SDFs) is well-suited to differentiable collision detection [2, 53]. This leads us to two requirements: i) we need a way to compute and represent an SDF based on a mesh; ii) since the true SDF surface is locally flat, we need a way of smoothing it. Grasp'D [2], addressed these requirements by pre-computing a discretized SDF grid from which values were computed by trilinear interpolation, which acts as a simple form of smoothing. Storing and querying the grid has a constant memory and compute cost. Surface normals can be computed (differentiably) by finite differencing. On the other hand, the grid is an approximation that loses details from the underlying mesh and under constant voxel resolutions is not cheaper to use with simpler meshes.

In this work, we compute the object SDF directly on the triangular, watertight mesh representing the object. We leverage the bounding volume hierarchy (BVH) data structure to accelerate the query of the closest mesh face to compute the distance, and use ray casting to determine the sign of the result (negative sign means inside, positive sign means outside the mesh). Since we compute the true SDF and not an approximation, local flatness of the mesh – i.e., constant normals along each face – creates zero-gradient plateaus that are hard to escape. To address this, we take inspiration from Phong tessellation [14], a rendering technique that smooths mesh normals and silhouettes, and computes a smoothed *Phong SDF*. This is more accurate than discretizing, requires $7\times$ fewer SDF queries (by avoiding finite differencing

for normals), and lets computation time scale with mesh complexity, leading to a large speedup when using simplified meshes provided by DDGC [3]. Discontinuities between faces – where normals vary sharply along an edge – create unstable regions where gradient steps produce unexpected results. Here we follow [2] and take a coarse-to-fine smoothing approach (see Section III-B).

Instability in integration and optimization – Our grasp synthesis method consists of an inner loop (integration) that computes a simulation-based grasp metric maximized by an outer loop (optimization). Study of the outer-loop optimization properties of different differentiable integration schemes is just beginning [54, 55]. Inner-loop instability may arise from simulating hard contact constraints with a force-based integrator. Enforcing non-penetration between rigid bodies under a force-based integrator requires high-stiffness constraints. Such constraints may cause instability, necessitating short time steps, since small pose changes (inducing small constraint violating interpenetrations) create large forces. Instability in the inner loop (integration) destabilizes the outer loop (optimization), creating a rugged loss landscape that is hard to optimize over, even with gradient information.

Our previous work [2] handled this instability with a problem relaxation that allowed (minimal) physics violations. Inspired by Contact-Invariant Optimization [37, 56], each contact point was assigned a corresponding six-dimensional variable representing the desired resultant object wrench. Instead, our current approach (see Section III-C) is built on position-based dynamics (PBD [57]), known for stability and robustness in contact-rich scenarios. A more stable inner loop allows grasp synthesis to succeed without introducing additional variables, leading to a significant speed increase (about $20\times$ over [2]).

Contact sparsity – Most points on the hand are not in contact with the object and will not be brought into contact by an infinitesimal pose perturbation. This means most hand vertices do not contribute to the simulator gradient, so it is difficult for gradient-descent to create new contacts. To address this, we adapt the *leaky* contact gradients of [2] to PBD.

B. Shape representation

Signed distance function (SDF) – Whereas primitive objects (e.g., a sphere or box) admit an analytic SDF, this is not the case for complex objects, for which an SDF representation is not readily available. We represent the object to be grasped by a mesh in canonical pose, from which we compute SDF values on the fly as $\phi(\mathbf{r}) = \mathbf{r}^{\pm}_{\text{obj}}\|\mathbf{r}-\mathbf{r}^*\|$, where $\mathbf{r}^*$ (the closest point on the mesh surface to $\mathbf{r}$) and $\mathbf{r}^{\pm}_{\text{obj}}$ (a positive/negative sign indicating whether $\mathbf{r}$ is outside/inside the mesh volume) are queried from Warp.

Phong SDF – As described in the introduction of this section, we use the smoothed *Phong SDF* $\rho(\mathbf{r})$ for contact generation [14]. The *Phong SDF* is the SDF of a quadratic surface matching normals interpolated from the vertices of the face $\mathbf{r}^*$ lies on according to barycentric coordinates. Say $\mathbf{r}^* = u\mathbf{v_i} + v\mathbf{v_j} + w\mathbf{v_k}$, with u, v, w being the barycentric coordinates and $\mathbf{v_i}, \mathbf{v_j}, \mathbf{v_k}$ being the vertices with vertex normals $\mathbf{n_i}, \mathbf{n_j}, \mathbf{n_k}$. We then define the closest point to $\mathbf{r}$ on the quadratic surface as

$$\mathbf{r}^*_\rho = (1-\alpha)\alpha(u,v,w)\begin{pmatrix}\psi_i(\mathbf{r}^*)\\ \psi_j(\mathbf{r}^*)\\ \psi_k(\mathbf{r}^*)\end{pmatrix},$$

where $\psi_i(\mathbf{r}^*)$ is the projection of $\mathbf{r}^*$ onto the plane defined by $\mathbf{v_i}$ (and $\mathbf{n_i}$), and α controls interpolation between the flat and curved triangle. Finally, we may write our *Phong SDF* as $\rho(\mathbf{r}) = \mathbf{r}^{\pm}_{\text{obj}}\|\mathbf{r}-\mathbf{r}^*_\rho\|$, with $\nabla\rho(\mathbf{r}) = (\mathbf{r}-\mathbf{r}^*_\rho)/\|\rho(\mathbf{r})\|$.

SDF dilation – In addition to *Phong SDF* smoothing, we follow the coarse-to-fine smoothing introduced in [2]. Specifically, we define the object surface *not* as the zero-level of the SDF ρ, but as the radius $r \geq 0$ level-set, which yields a padded, rounded version of the surface. We still want the final grasp to respect the real object geometry, so we decrease r to 0 on a linear schedule, gradually resolving the dilated SDF to the true surface as optimization continues.

C. Position-based dynamics

We represent the hand by its vertices, $\mathbf{p} = \text{FK}(\mathbf{q_h})$, whose positions are given by applying differentiable forward kinematics FK to the hand configuration $\mathbf{q_h}$. Each hand vertex $\mathbf{p_i}$ imposes a non-penetration constraint from which we compute positional updates to the object orientation (we treat the hand as static). Let $\mathbf{x}$, θ and $\dot{\mathbf{x}}$, $\dot{\theta}$, be the object position, orientation, and their time derivatives. We can express each hand point in terms of its location in the object local frame as $\mathbf{r_i} = R(\theta)^{-1}(\mathbf{p_i} - \mathbf{x})$, where $R(\theta)$ is the object rotation matrix. The non-penetration constraint can then be written as $C(\mathbf{p_i}) = \max(0, -\rho(\mathbf{r_i}))$, which computes penetration depth using the object *Phong SDF*. Given current values for $\mathbf{x}^-, \dot{\mathbf{x}}^-, \theta^-, \dot{\theta}^-$, to perform an integration step, we first compute predicted values with a symplectic Euler update (see preliminaries in [58]), yielding $\tilde{\mathbf{x}}, \tilde{\dot{\mathbf{x}}}, \tilde{\theta}, \tilde{\dot{\theta}}$. Next, we compute updates based on each constraint as:

$$\left[\nabla\mathbf{x}^T\nabla\theta\right]^T = -\mathbf{M}^{-1}\mathbf{J}_C^T\left[\mathbf{J}_C\mathbf{M}^{-1}\mathbf{J}_C^T\right]^{-1}C(\mathbf{p_i}), \quad (1)$$

where $\mathbf{M}$ is the mass matrix and $\mathbf{J}_C(\mathbf{x},\theta) = \left(\frac{\partial C}{\partial \mathbf{x}}\frac{\partial C}{\partial \theta}\right)$ is the constraint Jacobian. We perform a Gauss-Seidel step to update the predicted values for all constraints, yielding $\mathbf{x}^+, \theta^+$, and set derivatives accordingly as $\dot{\mathbf{x}}^+ = (\mathbf{x}^+ - \mathbf{x}^-)/\Delta t$ and $\dot{\theta}^+ = (\theta^+ - \theta^-)/\Delta t$, where Δt is the timestep length. We follow the Coulomb friction formulation from [58].

Leaky gradient – The non-penetration constraint shows that a hand vertex (say $\mathbf{p_i}$ and $\mathbf{r_i}$ in world and object frame) not in collision with the object (i.e., with $\rho(\mathbf{r_i}) > 0$) will have $C(\mathbf{p_i}) = 0$ and, from equation (1), will not contribute to the PBD update. A small perturbation of hand pose will not create contact ($\rho(\mathbf{r_i} + \epsilon) > 0$), so the vertex will not contribute to hand pose gradient. This makes it difficult to follow gradient to create new contacts. We address this with the *leaky gradient* introduced in [2]. Specifically, rather than using the correct gradient $\frac{\partial C}{\partial \mathbf{p}} = \begin{cases}\nabla\phi(\mathbf{p}) & \text{if } \phi(p) < 0\\ 0 & \text{otherwise}\end{cases}$, we use a biased gradient $\frac{\partial C}{\partial \mathbf{p}} = \begin{cases}\nabla\phi(\mathbf{p}) & \text{if } \phi(\mathbf{p}) < 0\\ \alpha\nabla\phi(\mathbf{p}) & \text{otherwise}\end{cases}$, where

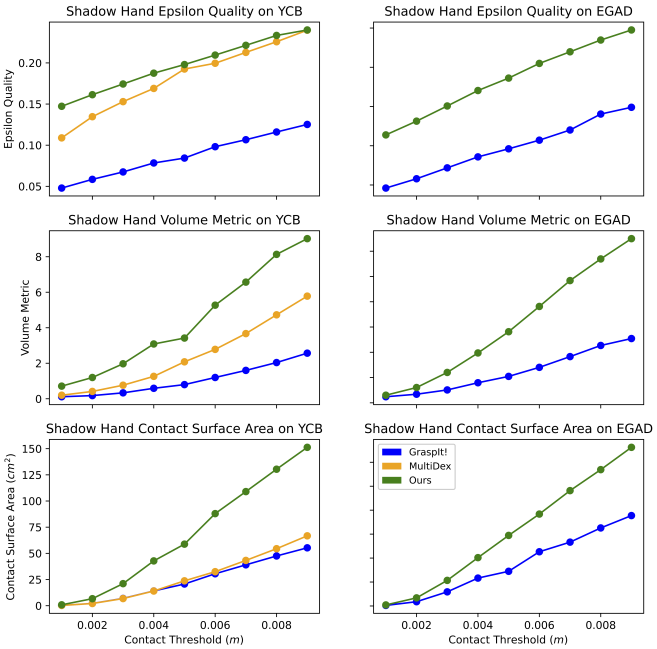

Fig. 3: Grasp metrics such as epsilon quality, GWS volume, and contact surface area depend on the threshold distance used for contact generation. Our method improves on GraspIt! [1] and MultiDex [50] baselines under all threshold choices. Results for the Barrett and Allegro hands (available on our website) follow a similar trend.

$\alpha \in [0, 1]$ controls how much gradient leaks through. We set $\alpha = 0.1$ in our experiments.

IV. Grasp'D-1M: Dexterous Grasp Dataset

Fast-Grasp'D enables an algorithmic pipeline for generating multi-finger grasps. Given object and gripper sets, we synthesize many grasps for each $(gripper, object)$ pair with the object set in canonical pose. We call these *base grasps*. Next, we generate several scenes for each object by dropping the object (in simulation) on a table and letting it come to rest in a natural pose. We transfer *base grasps* to scenes by applying the object pose transform to the gripper pose, yielding a larger set of *scene grasps*. Finally, we render each *scene* from multiple viewpoints, yielding a set of training examples that can be used to train vision-based grasping.

A. Gradient-based Grasp Optimization

Computing the grasp metric – To measure the quality of a candidate grasp $\mathbf{q_h}$, we test its ability to withstand forces applied to the object. Specifically, we set an initial object velocity $\dot{\mathbf{x}}^{(0)}$ and test whether contact with the static gripper can dampen it. The object is always initialized in canonical pose $(\mathbf{x}^{(0)}, \theta^{(0)})$. We simulate according to Section III-C and compute the object's final (translational and angular) velocity $(\dot{\mathbf{x}}^{(T)}, \dot{\theta}^{(T)})$. The more the velocity is dampened, the more stable we estimate the grasp to be. Of course, testing a single force is not sufficient and we perform a batch of M simulations in parallel, each setting a different initial object velocity $\dot{\mathbf{x}}_m^{(0)}$. In our experiments we use $M = 7$, setting positive and negative velocities along each axis as well as one simulation with $\dot{\mathbf{x}}^{(0)} = 0$. We simulate for a single timestep, with $\Delta t = 0.001s$. Our stability loss is then

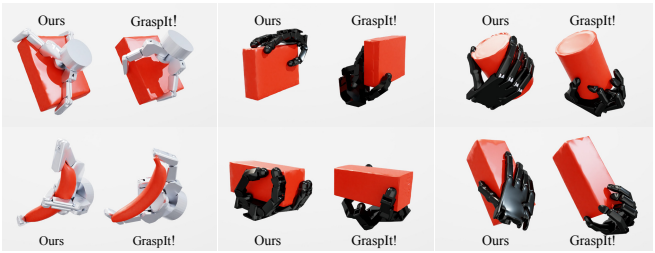

Fig. 4: Contact-rich grasps can be generated by our method which optimizes in the full DOF-space of the hand. The GraspIt! [1] planner mainly generates fingertip grasps. The baseline grasps exhibit fewer contacts that result in reduced stability compared to grasps synthesized by our method.

defined as

$$\mathcal{L}_{\text{stable}}(\mathbf{q_h}) = \sum_{m=1}^{M} \frac{\|\dot{\mathbf{x}}_m^{(T)}\| + \|\dot{\theta}_m^{(T)}\|}{M}. \tag{2}$$

Additional losses – $\mathcal{L}_{\text{range}}$ encourages hand joints to be near the middle of their ranges. $\mathcal{L}_{\text{limit}}$ penalizes hand joints outside of their range.

$$\mathcal{L}_{\text{range}}(\mathbf{q_h}) = \|\mathbf{q_h} - \frac{\mathbf{q_h}^{\text{up}} + \mathbf{q_h}^{\text{low}}}{2}\| \tag{3}$$

$$\mathcal{L}_{\text{limit}}(\mathbf{q_h}) = \max(\mathbf{q_h} - \mathbf{q_h}^{\text{up}}, 0) + \max(\mathbf{q_h}^{\text{low}} - \mathbf{q_h}, 0) \tag{4}$$

Optimization – We sample hand initializations $\mathbf{q_h}$ with the approach-sampling procedure described in [2]. We optimize using simple gradient descent with a learning rate of 0.001.

Dataset generation – We use the Omniverse Replicator to generate and render scenes as visual input for learning. Each scene has an object on a table. To generate the scene, we sample an object pose above the table and simulate dropping the object and letting it come to rest. The renders include RGB, depth, 2D and 3D object bounding boxes and a segmentation (separating the image into labelled regions for the table, object and background) in both mono and stereo. We generate training tuples by transferring base grasps to each scene. Specifically we apply the scene's object pose transform to the base grasp and check for interpenetration with the table.

V. Experiments

Datasets – *Grasp'D-Base* is our dataset of one million *base grasps* of canonically posed EGAD [59] and YCB [60] objects with the Barrett, Allegro, and Shadow hands. Our main dataset of multi-modal visual training examples, *Grasp'D-1M*, is generated by transferring these base grasps to randomized scenes and rendering as described in Section IV. In all experiments, we use the original DDGC [3] renders and scenes to match the original intended design parameters for a fair comparison. The dataset labels used in Tables II and III are explained below. *DDGC* refers to the Barrett hand dataset provided by [3]. *GraspIt!* refers to a set of baseline grasps for DDGC scenes generated with the GraspIt! [1] simulated annealing planner (for 70k steps for the Barrett and Allegro hands and 40k steps for the Shadow hand). *Ours* refers to a similar-sized set of grasps created by transferring grasps from

Objects	Hand	Dataset	CA↑	IV↓	ϵ ↑	Vol ↑	SD ↓
Scale (Unit)			cm^2	cm^3			cm
YCB	Barrett	*DDGC* [3]	4.64	2.73	0.10	0.26	2.74
		GraspIt! [1]	7.18	**0.54**	0.13	0.46	2.10
		MultiDex [50]	13.82	5.55	0.18	1.38	**1.24**
		Ours	**55.71**	4.70	**0.23**	**5.09**	1.40
	Allegro	*GraspIt!*	12.37	**1.80**	0.15	1.80	2.01
		MultiDex	21.16	6.61	**0.18**	2.19	1.93
		Ours	**49.17**	5.62	**0.18**	**3.42**	**1.66**
	Shadow	*GraspIt!*	20.72	6.68	0.08	0.80	2.88
		MultiDex	21.47	6.70	0.17	1.90	2.60
		Ours	**58.60**	**6.34**	**0.18**	**3.42**	**2.42**
EGAD	Barrett	*DDGC*	0.69	**0.02**	0.05	0.07	2.84
		GraspIt!	10.96	2.13	0.14	0.81	1.68
		Ours	**58.17**	4.80	**0.24**	**3.71**	**0.99**
	Allegro	*GraspIt!*	14.38	**1.98**	0.16	1.98	1.57
		Ours	**36.68**	6.54	**0.22**	**4.49**	**1.05**
	Shadow	*GraspIt!*	28.96	8.91	0.12	1.61	3.27
		Ours	**58.91**	**5.18**	**0.24**	**7.05**	**1.71**

TABLE II: Dataset comparison. Our method is able to find more contact-rich, stable grasps compared to the GraspIt! [1] baseline and the training set of DDGC [3]. Specifically, we discover grasps with higher contact area (**CA**) and stability, as measured by epsilon metric (ϵ), volume metric (**Vol**), and simulation displacement (**SD**). This results in somewhat higher interpenetration volume (**IV**), but maintains a similar ratio between intersection and contact area. All reported figures are top10 (ranked by epsilon metric).

Test Set	Train Set	CA↑	IV↓	ϵ ↑	Vol ↑	SD ↓
Scale (Unit)		cm^2	cm^3			cm
EGAD Val.	*DDGC* [3]	3.17	**12.37**	0.11	0.35	1.09
	GraspIt! [1]	3.30	13.62	0.15	0.71	1.48
	Ours	**5.02**	13.07	**0.18**	**0.85**	**0.60**
KIT	*DDGC*	4.28	**10.33**	0.09	0.25	1.00
	GraspIt!	4.31	13.07	0.13	0.42	1.55
	Ours	**4.60**	10.58	**0.18**	**0.81**	**0.86**
EGAD+KIT	*DDGC*	3.45	11.54	0.11	0.40	1.01
	GraspIt!	3.52	13.13	0.14	0.64	1.58
	Ours	**4.70**	**11.14**	**0.20**	**1.04**	**0.56**

TABLE III: Training vision-based grasping. Retraining the DDGC [3] network, which predicts Barrett hand grasps from RGBD and instance segmentation inputs, with data generated by our method results in predicted grasps with 30% more contact area (**CA**), 33% higher epsilon metric (ϵ) and 35% lower simulated displacement (**SD**).

Grasp'D-Base to the same DDGC scenes. *MultiDex* refers to the simulation filtered grasp dataset provided by [50] which we transfer to DDGC scenes in the same manner.

Metrics – Grasps (dataset and predictions) are evaluated with geometric metrics: (1) contact area (**CA**), (2) intersection volume (**IV**), (3) epsilon metric (ϵ), (4) grasp wrench space volume metric (**Vol**), and (5) simulation displacement (**SD**). Notably, **CA**, ϵ and **Vol** depend on the distance threshold used for contact generation. Previous works use varying thresholds (e.g., 9mm in [3] and 2mm in [61]). We plot these metrics for a wide range of thresholds (Figure 3) and show *Fast*-Grasp'D significantly outperforms baselines under all choices.

A. Grasp'D-1M Evaluation: Grasp Quality Metrics

Figure 3 shows that Shadow hand grasps generated by our method strictly dominate the GraspIt! and MultiDex baselines in terms of epsilon quality, GWS volume, and contact surface area under all contact thresholds, across two object sets (YCB and EGAD). Results for the Barrett and Allegro hands (available on our website) follow a similar trend. Figure 4 shows examples grasps from our method and GraspIt!. Whereas GraspIt! mainly discovers grasps with fingertip contact, we find high-contact grasps that conform to object surface geometry. Table II shows additional statistics with a medium contact threshold of 5mm.

B. RGBD Grasp Prediction with Grasp'D-1M

To confirm the generated data can in practice improve vision-based grasp pipelines, we retrain an existing network, DDGC [3], on data from our method or the GraspIt! baseline. For simplicity, we limit our evaluation to single-object scenes. We follow the training procedure described in [3] and find training converges after 3000 epochs. Table-III reports our results, which show training on our dataset results in predicted grasps with 30% more contact (**CA**), 33% higher epsilon metric (ϵ), and 35% lower simulated displacement (**SD**).

C. Computational Performance.

We achieve a roughly 10x speedup compared to the GraspIt! [1] simulated annealing planner and a 20x speedup compared to a previous differentiable grasp synthesis method [2]. The GraspIt! simulated annealing planner takes around 20s to generate a Barrett hand grasp. [2] takes about 5 minutes to generate a single grasp online or 20 seconds amortized over parallel generations in a batch. Our method can generate a contact-rich grasp of a YCB object in about 1s amortized, while online generation of a single grasp takes about 10s. Grasp'D and *Fast*-Grasp'D timings are reported with a batch size of 32 on an NVIDIA A100.

VI. Conclusion

In this work, we have introduced a new method to synthesize multi-fingered grasps that leverages our differentiable grasping simulator to achieve improved stability and contact richness compared to commonly used baselines. Our experiments have demonstrated the benefits of our approach to an existing multi-finger robotic grasping pipeline that we trained on our generated grasps. With the release of a large-scale dataset of high-quality grasps synthesized through our method for various robotic hands, we aim to further advance the state of the art in multi-fingered robotic grasping.

VII. Acknowledgements

The authors gratefully acknowledge the support of NSERC, Vector, CIFAR, UofT, Nvidia and Samsung.

REFERENCES

[1] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, 2004.

[2] D. Turpin, L. Wang, E. Heiden, Y.-C. Chen, M. Macklin, S. Tsogkas, S. Dickinson, and A. Garg, "Grasp'D: Differentiable Contact-rich Grasp Synthesis for Multi-fingered Hands," in *European Conference on Computer Vision*, 2022.

[3] J. Lundell, F. Verdoja, and V. Kyrki, "DDGC: Generative deep dexterous grasping in clutter," *IEEE Robotics and Automation Letters*, 2021.

[4] W. Wei, D. Li, P. Wang, Y. Li, W. Li, Y. Luo, and J. Zhong, "DVGG: Deep variational grasp generation for dextrous manipulation," *IEEE Robotics and Automation Letters*, 2022.

[5] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic, *et al.*, "Deep Learning Approaches to Grasp Synthesis: A Review," *arXiv:2207.02556*, 2022.

[6] S. Brahmbhatt, C. Ham, C. C. Kemp, and J. Hays, "Contactdb: Analyzing and predicting grasp contact via thermal imaging," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[7] S. Brahmbhatt, C. Tang, C. D. Twigg, C. C. Kemp, and J. Hays, "ContactPose: A dataset of grasps with object contact and hand pose," in *European Conference on Computer Vision*, 2020.

[8] S. Brahmbhatt, A. Handa, J. Hays, and D. Fox, "Contactgrasp: Functional multi-finger grasp synthesis from contact," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[9] T. Geng, M. Lee, and M. Hülse, "Transferring human grasping synergies to a robot," *Mechatronics*, 2011.

[10] A. Lakshmipathy, D. Bauer, C. Bauer, and N. S. Pollard, "Contact transfer: A direct, user-driven method for human to robot transfer of grasps and manipulations," in *IEEE International Conference on Robotics and Automation*, 2022.

[11] J. Lundell, E. Corona, T. N. Le, F. Verdoja, P. Weinzaepfel, G. Rogez, F. Moreno-Noguer, and V. Kyrki, "Multi-fingan: Generative coarse-to-fine sampling of multi-finger grasps," in *IEEE International Conference on Robotics and Automation*, 2021.

[12] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *IEEE International Conference on Robotics and Automation*, 2009.

[13] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, 2012.

[14] T. Boubekeur and M. Alexa, "Phong tessellation," in *ACM SIGGRAPH Asia 2008 papers*, 2008.

[15] H. Zhang, J. Tang, S. Sun, and X. Lan, "Robotic Grasping from Classical to Modern: A Survey," *arXiv:2202.03631*, 2022.

[16] E. Corona, A. Pumarola, G. Alenya, F. Moreno-Noguer, and G. Rogez, "GANHand: Predicting human grasp affordances in multi-object scenes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[17] K. Karunratanakul, J. Yang, Y. Zhang, M. J. Black, K. Muandet, and S. Tang, "Grasping field: Learning implicit representations for human grasps," in *International Conference on 3D Vision*, 2020.

[18] N. Khargonkar, N. Song, Z. Xu, B. Prabhakaran, and Y. Xiang, "NeuralGrasps: Learning Implicit Representations for Grasps of Multiple Robotic Hands," in *Conference on Robot Learning*, 2022.

[19] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, 2019.

[20] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *International Conference on Computer Vision*, 2019.

[21] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *IEEE International Conference on Robotics and Automation*, 2021.

[22] Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, and Y. Zhu, "Synergies between affordance and geometry: 6-dof grasp detection via implicit representations," in *Robotics: Science and Systems*, 2021.

[23] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *IEEE International Conference on Robotics and Automation*, 2015.

[24] C. Eppner, A. Mousavian, and D. Fox, "ACRONYM: A large-scale grasp dataset based on simulation," in *IEEE International Conference on Robotics and Automation*, 2021.

[25] M. Danielczuk, J. Xu, J. Mahler, M. Matl, N. Chentanez, and K. Goldberg, "Reach: Reducing false negatives in robot grasp planning with a robust efficient area contact hypothesis model," in *International Symposium of Robotic Research*, 2019.

[26] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid, "Learning joint reconstruction of hands and manipulated objects," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[27] B. Doosti, S. Naha, M. Mirbagheri, and D. J. Crandall, "Hope-net: A graph-based model for hand-object pose estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[28] H. Jiang, S. Liu, J. Wang, and X. Wang, "Hand-object contact consistency reasoning for human grasps generation," in *International Conference on Computer Vision*, 2021.

[29] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, "Unigrasp: Learning a unified model to grasp with multifingered robotic hands," *IEEE Robotics and Automation Letters*, 2020.

[30] F. T. Pokorny and D. Kragic, "Classical grasp quality evaluation: New algorithms and theory," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[31] C. Ferrari and J. F. Canny, "Planning optimal grasps." in *IEEE International Conference on Robotics and Automation*, 1992.

[32] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *IEEE International Conference on Robotics and Automation*, 2012.

[33] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems*, 2017.

[34] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *International Journal of Robotics Research*, 2012.

[35] J. Mahler, F. T. Pokorny, Z. McCarthy, A. F. van der Stappen, and K. Goldberg, "Energy-bounded caging: Formal definition and 2-D energy lower bound algorithm based on weighted alpha shapes," *IEEE Robotics and Automation Letters*, 2016.

[36] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[37] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem," in *Robotics: Science and Systems Manipulation Workshop - Sensing and Adapting to the Real World*, 2007.

[38] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics," in *IEEE International Conference on Robotics and Automation*, 2019.

[39] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, "DiffTaichi: Differentiable Programming for Physical Simulation," in *International Conference on Learning*

Representations, 2020.

[40] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "ADD: Analytically Differentiable Dynamics for Multi-Body Systems with Frictional Contact," *ACM Transactions on Graphics*, 2020.

[41] 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," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[42] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics for articulated rigid bodies with contact," in *Robotics: Science and Systems*, 2021.

[43] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin, "Efficient differentiable simulation of articulated bodies," in *International Conference on Machine Learning*, 2021.

[44] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "NeuralSim: Augmenting Differentiable Simulators with Neural Networks," in *IEEE International Conference on Robotics and Automation*, 2021.

[45] E. Heiden, M. Macklin, Y. S. Narang, D. Fox, A. Garg, and F. Ramos, "DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting," in *Robotics: Science and Systems*, 2021.

[46] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin, "Accelerated Policy Learning with Parallel Differentiable Simulation," in *International Conference on Learning Representations*, 2022.

[47] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, "Deep differentiable grasp planner for high-dof grippers," in *Robotics: Science and Systems*, 2020.

[48] T. Liu, Z. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu, "Synthesizing Diverse and Physically Stable Grasps With Arbitrary Hand Structures Using Differentiable Force Closure Estimator," *IEEE Robotics and Automation Letters*, 2021.

[49] J. Romero, D. Tzionas, and M. J. Black, "Embodied Hands: Modeling and Capturing Hands and Bodies Together," *ACM Transactions on Graphics*, 2017.

[50] P. Li, T. Liu, Y. Li, Y. Zhu, Y. Yang, and S. Huang, "GenDexGrasp: Generalizable Dexterous Grasping," *arXiv:2210.00722*, 2022.

[51] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang, "DexGraspNet: A Large-Scale Robotic Dexterous Grasp Dataset for General Objects Based on Simulation," *arXiv:2210.02697*, 2022.

[52] M. Macklin, "WARP: A High-performance Python Framework for GPU Simulation and Graphics," March 2022, nVIDIA GPU Technology Conference (GTC).

[53] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and Z. Corse, "Local optimization for robust signed distance field collision," *ACM on Computer Graphics and Interactive Techniques*, 2020.

[54] H. J. Suh, M. Simchowitz, K. Zhang, and R. Tedrake, "Do differentiable simulators give better policy gradients?" in *International Conference on Machine Learning*, 2022.

[55] Y. D. Zhong, J. Han, and G. O. Brikis, "Differentiable Physics Simulations with Contacts: Do They Have Correct Gradients wrt Position, Velocity and Control?" *arXiv:2207.05060*, 2022.

[56] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *International Journal of Robotics Research*, 2009.

[57] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, 2007.

[58] C. Deul, P. Charrier, and J. Bender, "Position-based rigid-body dynamics," *Computer Animation and Virtual Worlds*, 2016.

[59] D. Morrison, P. Corke, and J. Leitner, "EGAD! An Evolved Grasping Analysis Dataset for Diversity and Reproducibility in Robotic Manipulation," *IEEE Robotics and Automation Letters*, 2020.

[60] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *International Journal of Robotics Research*, 2017.

[61] P. Grady, C. Tang, C. D. Twigg, M. Vo, S. Brahmbhatt, and C. C. Kemp, "Contactopt: Optimizing contact to improve grasps," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.