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

DefGraspNets: Grasp Planning on 3D Fields with Graph Neural Nets

Isabella Huang[1], Yashraj Narang[2], Ruzena Bajcsy[1], Fabio Ramos[2,3], Tucker Hermans[2,4], Dieter Fox[2,5]

***Abstract*— Robotic grasping of 3D deformable objects is critical for real-world applications such as food handling and robotic surgery. Unlike rigid and articulated objects, 3D deformable objects have infinite degrees of freedom. Fully defining their state requires 3D deformation and stress fields, which are exceptionally difficult to analytically compute or experimentally measure. Thus, evaluating grasp candidates for grasp planning typically requires accurate, but slow 3D finite element method (FEM) simulation. Sampling-based grasp planning is often impractical, as it requires evaluation of a large number of grasp candidates. Gradient-based grasp planning can be more efficient, but requires a differentiable model to synthesize optimal grasps from initial candidates. Differentiable FEM simulators may fill this role, but are typically no faster than standard FEM. In this work, we propose learning a predictive graph neural network (GNN), DefGraspNets, to act as our differentiable model. We train DefGraspNets to predict 3D stress and deformation fields based on FEM-based grasp simulations. DefGraspNets not only runs up to 1500x faster than the FEM simulator, but also enables fast gradient-based grasp optimization over 3D stress and deformation metrics. We design DefGraspNets to align with real-world grasp planning practices and demonstrate generalization across multiple test sets, including real-world experiments.**

I. Introduction

Deformable objects are omnipresent in our world, and grasping them is critical for food handling [1], robotic surgery [2], and domestic tasks [3,4]. However, their physical complexities pose challenges for key aspects of grasp planning, including modeling, simulation, learning, and optimization. Deformable objects have infinite degrees of freedom and require continuum mechanics models to accurately predict their responses to body forces (e.g., gravity) and surface tractions (e.g., contacts). For deformable solids, continuum models can predict two field quantities critical for robot grasping, *stress* tensors and *deformation* vectors defined at every point in the object [5]. In general, low-stress grasps are desirable to reduce material fatigue from repeated grasping, or to avoid exceeding the yield stress of the object, at which point permanent deformation or failure occurs. Predicting deformation is also critical, especially when grasping containers. One may want to minimize the deformation on a box of crackers to avoid crushing the contents, or maximize the deformation on a bottle of ketchup to efficiently squeeze out the contents.

[1]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA ;[2]NVIDIA Corporation, Seattle, USA;[3]School of Computer Science, University of Sydney, Sydney, Australia;[4]School of Computing, University of Utah, Salt Lake City, USA;[5]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA

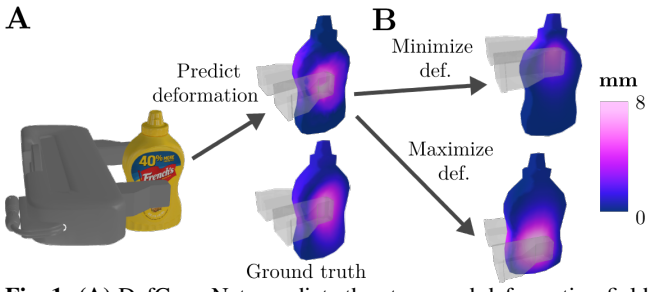


Fig. 1: (A) DefGraspNets predicts the stress and deformation fields from grasping an unseen object 1500x faster than FEM, and **(B)** enables gradient-based grasp refinement to optimize these fields.

Although knowledge of stress and deformation fields is useful, deriving closed-form solutions is intractable for general cases. Moreover, direct real-world measurement is extremely difficult without cumbersome instrumentation. Consequently, robotic grasping has historically leveraged rigid-body models, for which deformation is ignored and object state can be simply described by 6D pose and velocity [6,7].

On the other hand, we can use deformable-object *simulators* to access these quantities and plan grasps accordingly. However, such simulators rely on complex numerical models like the gold-standard 3D finite element method (FEM) [8,9]. Although FEM can simulate the result of any grasp on a deformable object [10], each evaluation can take minutes on a CPU-based industry-standard simulator [11] and seconds on a GPU-based robotics simulator [12], which is prohibitively slow for online grasp planning. Moreover, while differentiable simulators enable gradient-based optimization for parameter estimation [13] and control optimization [13,14], few studies have explored their application to robotic grasping, and they are typically slower than standard FEM.

We propose *DefGraspNets*, a graph neural network that can enable grasp planning by predicting the stress and deformation fields resulting from grasps and allowing efficient optimization (Fig. 1). We demonstrate that this network is 1) **fast**, with a ∼1500x speed-up compared to a GPU-based FEM simulator, 2) **accurate**, with stress and deformation fields consistent with ground-truth FEM, 3) **generalizable**, with reliable rankings of grasp candidates over unseen poses, elastic moduli, in-category objects, and out-of-category objects, and 4) **differentiable**, enabling gradient-based optimization for grasp refinement. Finally, we conduct pilot studies that verify agreement of DefGraspNets, trained purely in simulation, with real-world outcomes. Data and code can be found on our website[1].

[1]https://sites.google.com/view/defgraspnets

II. Related Work

Grasp planning has received significant attention in robotics [6,15–18]. Recent works leverage learning-based approaches to enable fast planning and generalization to novel objects [19–23]. We focus on grasp planning for 3D deformable objects. Unlike rope or cloth, 3D deformables have dimensions of a similar magnitude along all 3 spatial axes and can undergo significant deformations along any of them [10]. We review grasp planning for 3D deformables, as well as methods for predicting stress and deformation fields via graph neural networks and differentiable simulation.

A. Grasp planning for deformable objects

Early works in grasp planning for deformable objects focused on finding *stable* grasps of planar objects, under which the object's strain energy would be maximized without inducing plastic deformation [24,25]. This has since been extended to the 3D case, where novel time-dependent grasp quality metrics have been proposed to capture the evolution of contact states under deformation [26,27].

Grasp planning for *deformation* of thin-walled containers (e.g., boxes, bottles) has also been explored. Given a 3D geometric stiffness map of the object, a minimal deformation grasp can be planned by localizing contact at high-stiffness regions. This map can be generated in simulation, via real-world probing [28], or from 2D images of the object via generative adversarial networks [29]. Grasp planning for *stress* has also been demonstrated via simulation on quasi-rigid objects using the boundary element method [30]. Finally, grasp planning for additional metrics can be performed with DefGraspSim, a 3D FEM-based grasp simulation framework [10]. For every grasp, it evaluates success, stability, stress, deformation, strain energy, and controllability.

These methods vary not only in the planning metric, but also in the type of computation required. Some require FEM simulation of the beginning of the interaction (e.g., just past initial contact) [25–28] or the full interaction [10,30], whereas others use neural networks [29]. Yet, all of these planners can only evaluate or predict the outcome of a candidate grasp, and cannot optimize grasps through gradient-based methods.

B. Graph neural networks for deformable-object interaction

Graph neural networks (GNNs) have been used to efficiently learn dynamics models for granular solids, deformable solids, and fluids [31–35]. Inspiring our work, MeshGraphNets [34] used GNNs to learn accurate dynamics for deformable solids using mesh-based representations, training from an industry-standard FEM solver. It predicted deformation and stress on a 3D deformable plate with kinematically-actuated colliding shapes and achieved evaluation speeds up to two orders of magnitude faster than the solver. RoboCraft [35] used GNNs to learn how plasticine-like objects with particle representations deform under interaction with a robotic gripper, training from visual input. Whereas MeshGraphNets used forward passes through the networks to predict dynamics, RoboCraft also used backwards passes to perform gradient-based trajectory optimization, molding the plasticine into a desired shape.

Our work also utilizes a GNN as a surrogate simulator for dynamics predictions. Unlike MeshGraphNets, which uses N-step rollouts to predict a final state via intermediate steps, DefGraspNets performs direct, one-step predictions of the final state. One-step prediction ensures that gradients are only propagated once through the network rather than over tens or hundreds of steps, mitigating vanishing or exploding gradients [36]. Furthermore, we focus on quasistatic rather than dynamic grasping; the ability of multi-step rollouts to predict object and controller dynamics offers limited advantage. Our ablation study verifies that using single-step predictions in our setting performs better than multi-step predictions (c.f. Sec. VIII).

Like RoboCraft, we design our network to include gripper actions in order to perform gradient-based optimization for grasp planning. Unlike RoboCraft and MeshGraphNets, we use force rather than position commands for our actuators, as force commands are implemented in notable industrial grippers [37–39] and are preferable for grasping (as opposed to applications like shape control). Gripper force determines whether the grasp will overcome the object's gravity, and gripper position cannot indicate force without additional knowledge (e.g., contact area, object stiffness). In addition, when grasping stiffer objects, position commands can induce high torques that can damage both the object and gripper. We also generalize our network to different elastic moduli, which was not explored in prior works.

C. Differentiable simulators

Differentiable simulators for rigid and deformable bodies allow gradients of output variables (e.g., poses, velocities, or deformation fields of objects) to be computed with respect to input variables (e.g., control inputs or material parameters) [40–46]. Such simulators enable gradient-based optimization for control optimization [14,45–48], parameter estimation [13,46,48], and inverse design [46,49].

There are 4 main strategies to realize a differentiable simulator or equivalent model: 1) finite-differencing a non-differentiable simulator, which has unfavorable $\mathcal{O}(n)$ scaling to an n-dimensional input space [50,51], 2) analytically or automatically differentiating a simulator that smoothly approximates spatial or kinetic discontinuities (e.g., penalty-based contact forces and smooth friction models [13,48], which may introduce inaccuracies or require tuning), 3) training a deep network with physically-based loss functions [52,53], which has seen limited use for contact dynamics [54], and 4) training a deep network on datasets from a non-differentiable simulator, primarily with graph-based inductive biases [31,32,34,55].

For our application, we aim to simulate robotic grasping of 3D deformable objects. Thus, we focus on gold-standard 3D FEM simulation of deformable objects with contact. For this application, strategy 2 has been explored in a handful of recent works [13], including differentiable projective dynamics [56,57]. However, such simulators typically execute

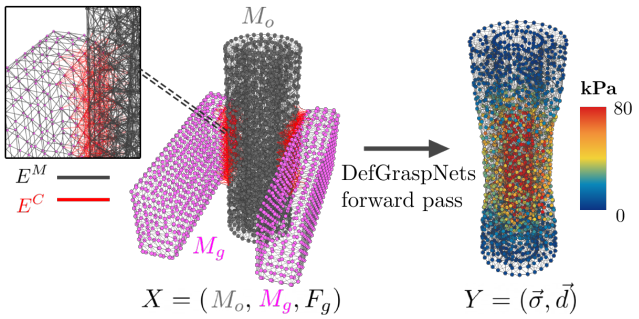


Fig. 2: Given a candidate grasp state X consisting of an object mesh M_o, gripper mesh M_g, and grasp force F_g, DefGraspNets generates contact edges E^C and predicts output Y consisting of a stress field $\vec{\sigma}$ and deformation field $\vec{d}$ defined at each node of the object mesh.

substantially slower than real-time (especially including a backward pass), and only one has realized differentiable FEM and contact modeled via the full nonlinear complementarity problem (NCP) [58] with both static and dynamic friction [57].

In this work, we explore strategy 4, training for the first time on a GPU-accelerated robotics FEM simulator [59] that addresses the full NCP [60] and has been experimentally validated across multiple studies [10–12]. To our knowledge, this effort also comprises the first application of such methods to robotic grasping of 3D deformable objects.

Strategy 2 has often been favored over strategy 4 due to the former's potential for generalizing to arbitrary physics [56,57]. Nevertheless, we show for the first time that strategy 4, through judicious selection and scaling of training data, can indeed generalize to novel grasps, elastic moduli, in-category objects, and out-of-category objects. Furthermore, the trained networks can execute 2 to 3 orders of magnitude faster than the reference simulator (i.e., faster than real-time).

III. The DefGraspNets Model

Here, we explain the GNN structure of DefGraspNets, including the input and output representations. We detail training data generation in Sec. IV and explain how we use DefGraspNets within a grasp planning algorithm in Sec. V.

A. Summary of inputs and outputs

DefGraspNets takes as input a *candidate grasp state* $X = (M_o, M_g, F_g)$ comprising a mesh M_o of a deformable object in its pre-contact state, a mesh M_g of the gripper fingers upon initial contact[2], and a total normal grasp force scalar F_g. A mesh is a collection of vertices and undirected edges that connect them. For M_o, these vertices and edges form tetrahedral elements that define the volumetric geometry of the object. For M_g, the vertices and edges form triangular elements that define the surface geometries of the fingers.

DefGraspNets converts the candidate grasp state X into a multigraph G (Sec. III-B), mapping the gripper-object contact interactions onto a graph structure. The multigraph is fed into an *Encode-Process-Decode* sequence [32,34,55](Sec. III-C). DefGraspNets predicts the stress and deformation $Y = (\vec{\sigma}, \vec{d})$ at steady state, at all vertices of M_o (Fig. 2). Please refer to Sec. IV for the formal definitions of these fields.

B. Multigraph representation

The multigraph representation $G = (V, E^M, E^C)$ has nodes V and undirected edge sets E^M and E^C; each edge stores the indices of its two connected nodes. We list their features, and mark those that differ from [34] with a ⋆ bullet.

Nodes. The nodes V correspond to the vertices of M_o and M_g. Each node v_i has a feature vector consisting of

- A 3-element one-hot vector for node type (i.e., part of M_g, M_o surface, or M_o interior)
- The 3D Cartesian position of the node
- ⋆ A 3D unit vector in the gripper closing direction. This is nonzero only for gripper nodes and informs the network which direction the grippers are closing.

Mesh edges. The mesh edges E^M correspond to the edges of M_o and M_g. Each mesh edge e^M_{ij} connects nodes v_i and v_j of the same type. Its feature vector consists of

- The 3D Cartesian displacement vector from v_i to v_j
- The scalar Euclidean distance between v_i and v_j
- ⋆ The scalar elastic modulus E of the deformable object. This is nonzero only for edges belonging to the object.

Contact edges. The contact edges E^C connect object and gripper nodes and are computed based on proximity at initial contact. Each edge e^C_{ij} is formed between a pair of nodes v_i and v_j that have different node types and are closer than hyperparameter ϵ. The edge's feature vector comprises

- The 3D Cartesian displacement vector from v_i to v_j
- The scalar Euclidean distance between v_i and v_j
- ⋆ The normalized grasp force F^C_g, which is the total grasp force F_g divided by the number of contact edges $|E^C|$.

C. Encoder, processor, & decoder architectures

First, all feature vectors associated with the nodes V, mesh edges E^M, and contact edges E^C are encoded into a common latent space with 3 respective multilayer perceptrons (MLPs). Then, L message-passing blocks with 3 separate MLPs per block sequentially aggregate and process information from adjacent nodes and edges. Finally, a decoder MLP takes the processed nodal features in the latent space and jointly outputs the predicted stress and Cartesian displacement per node in real units (Pa and m). Full details of the Encode-Process-Decode sequence can be found in [34].

IV. Data Generation and Model Training

We now describe our simulation-based approach to training DefGraspNets. We design a set of 60 object primitive models as a high-level abstraction of real-world geometries grouped into geometric categories (e.g., cuboids, cylinders, ellipsoids, annuli), and instances within each category have different dimensions and aspect ratios. Our dataset also includes a set of 11 of fruits and vegetables (e.g., apples, eggplants, potatoes) based on 3D scans [61]. Tetrahedral

[2]Although M_g comprises two unconnected parts, we refer to it collectively as the "gripper mesh."

volume meshes are generated for each deformable object using fTetWild [62]. Triangular surface meshes are generated for the gripper fingers using Onshape.

For each pre-contacted object mesh M_o, 100 grasps are generated using an antipodal sampler [63] wherein randomly-sampled surface points define gripper contact points, surface normals define grasp axes, and 4 rotations are regularly drawn about each grasp axis. These 100 grasps correspond to 100 gripper meshes M_g. Each grasp is evaluated using the DefGraspSim[10] simulation framework (built upon Isaac Gym[59] and the FleX FEM solver[60]) with the Franka parallel-jaw gripper. DefGraspSim evaluates the stress and deformation fields of the deformable object during grasping.

Given an object-grasp pair (M_o, M_g) in DefGraspSim, the gripper applies a linearly increasing amount of force on the object until $F_g^{max} = 15$N is reached in a zero-gravity environment.[3] This force was achieved by directly commanding DOF torque applied at the gripper joints. The values of the stress ($\vec{\sigma}$) and deformation fields ($\vec{d}$) at all object vertices are saved over 50 evenly-spaced substeps throughout the entire grasping trajectory. Formally, our dataset D is composed of input-output pairs, each consisting of a candidate grasp pose X_i and corresponding set of fields Y_i, that is, $D = \{X_i = (M_g, M_o, F_g), Y_i = (\vec{\sigma}, \vec{d})\}_{i=1}^N$, where $0 \leq F_g \leq 15$. Dataset D has $N = \#\ objects \times 100 \times 50 = 3.55e5$ unique points. Because our network performs one-step predictions of the final state and is ideal for quasistatic interactions, all unstable grasps involving chaotic dynamics are not included in D.

The values of the stress field $\vec{\sigma}$ at all object vertices are computed as follows: first, the second-order stress tensor at each tetrahedral element of M_o is acquired from DefGraspSim. The stress tensors at each vertex are calculated by averaging the stress tensors at all adjacent elements. Each stress tensor is then converted to the scalar von Mises stress (i.e., the second invariant of the deviatoric stress), which is widely used to quantify whether a material has yielded [5]. The values of the deformation field $\vec{d}$ are defined simply as the distance between the positions of the pre-contacted vertices of M_o and their positions under gripper force F_g.

Contact edges E^C are formed based on the threshold $\epsilon = 5$mm. Our networks are trained with a decaying learning rate from $5e^{-5}$ to $1e^{-6}$ over 25 epochs and a batch size of 1. A latent size of 128 and $L = 15$ message passing steps are used, where all MLPs have 2 hidden layers. Loss is defined as the sum of the MSE of stress and deformation over all nodes. On a single RTX 3090 GPU, the network trains at approximately 1600 steps per minute.

V. Grasp Planning

We demonstrate DefGraspNets as a grasp planner, where both gradient-free (i.e., evaluation of sampled grasps) and gradient-based refinement methods can be used to find an optimal grasp. We define Q as the optimization objective, which is any backwards pass-differentiable measure of the predicted deformation and/or stress fields (e.g., mean deformation, smooth differentiable approximation of maximum stress implemented in modern deep learning libraries).

A. Evaluation of sampled grasps

First, DefGraspNets supports online sampling-based grasp planning. For an unseen object, forward passes of DefGraspNets can be used to evaluate Q for 100 random antipodal grasps with parallel batches of size 5 in 7.3 seconds. In comparison, DefGraspSim requires approximately 3 hours to evaluate 100 grasps, which is $\sim$1500x slower.

The best grasp pose is identified as $T^* = \arg\min_{T \in T_s} Q(T; M_o)$, where T is a 6D rigid transformation applied to a constant initial state of the gripper M_g^0 wherein both fingers are maximally open. Any valid M_g can be fully defined by T and joint states $\vec{p}_g \in \mathbb{R}^2$ that determine how much each finger closes in order to contact M_o. These joint states $\vec{p}_g$ are calculated analytically by projecting the vertices of M_o onto the gripper faces, backprojecting the vertices within each face, and computing the minimum perpendicular distance over these vertices (i.e., the minimum contact distance) per finger.

B. Grasp refinement

Unlike existing deformable object planners, DefGraspNets' differentiability enables gradient-based refinement of a grasp pose to optimize Q. Starting from an initial grasp pose T_{init}, we perform gradient updates in the direction of $\partial Q / \partial T$ to achieve a refined T using backtracking line search [64] and simulated annealing [65]. With 12 refinement steps per grasp, refining 100 initial grasps requires approximately 8 minutes. A comparable time does not exist for DefGraspSim, as it is not differentiable.

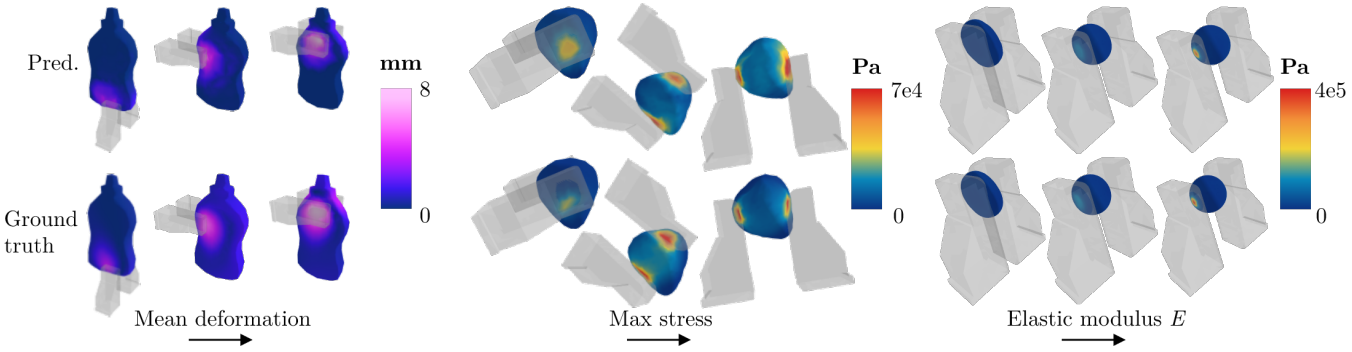
VI. Prediction Results

We test DefGraspNets' predictions of the ranking of grasps with respect to their mean stress and deformation values by quantifying the respective Kendall's τ rank correlation coefficients (τ_s and τ_d).[4] We answer the following questions for 4 levels of generalization:

1) Can DefGraspNets rank unseen grasps when trained on other grasps on the same object? (Ans: Yes. For an 80-20 train-test split over grasps on the same object, we get an average $\tau_s = 0.78$ and $\tau_d = 0.66$ over 15000 unseen X_i.)
2) Can DefGraspNets generalize to unseen elastic moduli E on the same object? (Ans: Yes. For a 7-3 train-test split over unique E for grasps on the same object, we get an average $\tau_s = 0.81$ and $\tau_d = 0.72$ over 15000 unseen X_i.)
3) Can DefGraspNets generalize to unseen primitive objects within the same geometric category? (Ans: Yes. For a 5-1 train-test split over unique objects, we get an average $\tau_s = 0.48$ and $\tau_d = 0.54$ over 15000 unseen X_i.)
4) Can DefGraspNets generalize to unseen real-world objects? (Ans: Yes. Moreover, we generate useful predictions

[3]For the elastic moduli examined ($1e^4 \leq E \leq 1e^7$ Pa), 15N was observed to induce substantial stress and deformation; gravity was ignored due to having negligible effect on stress and deformation compared to contact forces.

[4]Kendall's τ was chosen over Spearman's ρ for its comparative robustness (i.e., smaller gross error sensitivity).



(a) Mustard bottle, $F_g = 12$N, $E = 1e7$ **(b)** Strawberry, $F_g = 6$N, $E = 5e4$ **(c)** Sphere, $F_g = 5$N, $E \in [5e5, 1e6, 5e6]$

Fig. 3: A) Predicted and ground-truth deformation fields for a mustard bottle subject to grasps inducing increasing mean deformation, B) Predicted and ground-truth stress fields for a strawberry subject to grasps inducing increasing maximum stress, and C) Predicted and ground-truth stress fields for a sphere of increasing elastic moduli subject to identical grasps (deformation can be seen in resulting shape).

even when training on a small number of objects, as long as the train geometries are relevant to the test geometry as quantified by a low Chamfer distance. See Table I, which also reports the mean absolute error (*MAE*)).

Full visualizations of predicted field quantities for the 4th (i.e., most challenging) generalization level is shown in Fig. 3 on an unseen mustard bottle and unseen strawberry, as well as for the 2nd generalization level on a sphere.

VII. Grasp Planning Results

We demonstrate DefGraspNets as a grasp planner on 3 unseen objects (a mustard bottle, a lemon, and a strawberry) from existing datasets [66,67] with real-world elastic moduli. First, we perform evaluation of sampled grasps. On each unseen object, 100 random grasps T_r are generated, and the optimization metric $Q(T)$ is evaluated for each $T \in T_r$ via the forward pass of DefGraspNets. Of the 100 grasps, we select the 10 grasps that are predicted to yield the lowest Q ("threshold low" grasps), as well as 10 grasps that are predicted to yield the highest Q ("threshold high" grasps). We also randomly select 10 other grasps from the remaining 80 grasp candidates as a baseline. These 30 grasps are then evaluated within the ground-truth simulator DefGraspSim.

DefGraspNets is a reliable predictor of minimal- and maximal-Q grasps on the unseen objects, with 88% of these threshold-low and high grasps belonging to the set of 30 lowest and highest ground-truth-Q grasps, respectively.

Subsequently, we perform gradient-based grasp refinement on the threshold-low and threshold-high grasps to further reduce and increase Q, respectively. For each object, box plots in Fig. 4 visualize the distribution of ground-truth Q values for 5 groups of grasps: all sampled grasps, threshold-low grasps, threshold-low grasps after refinement, threshold-high grasps, and threshold-high grasps after refinement. In all cases, not only do threshold-high and low grasps from DefGraspNets yield substantially different ground-truth Q values, but refinement increases their polarity as desired.

The highest- and lowest-Q grasps generated by the sample-and-refine grasp planning procedure are shown in Fig. 5. These grasps align with physical reasoning (e.g., the highest-deformation grasps on the bottle and lemon compress

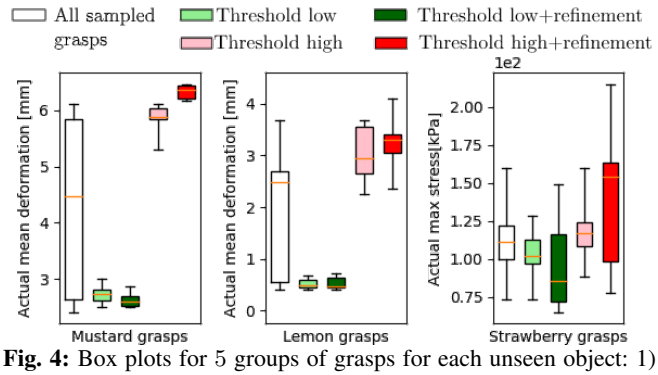


Fig. 4: Box plots for 5 groups of grasps for each unseen object: 1) all grasps, 2) threshold low grasps from sampling only, 3) threshold low grasps after refinement, 4) threshold high grasps from sampling only, and 5) threshold high grasps after refinement. The y-axis is the ground-truth Q value of these grasps as computed in DefGraspSim.

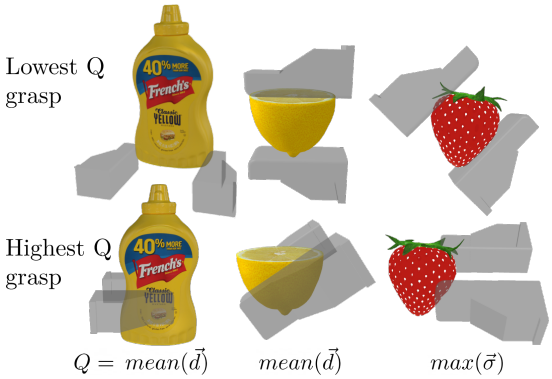


Fig. 5: Highest- and lowest-Q grasps for the mustard bottle, lemon, and strawberry generated by the sample-and-refine procedure.

the directions of lowest geometric stiffness; the highest-stress grasp on a strawberry concentrates force on minimal area). These grasps are also validated in the real world in Fig. 6.

VIII. Ablation Studies

We run several ablation studies on our network architecture design. Table II lists key design variables in DefGraspNets, with our selected conditions in bold. We compare our baseline model with 5 other trained models, each of which differ from baseline by exactly one condition. We compare performance on a fixed test set and report the Kendall's τ

TABLE I: Generalization to unseen real-world objects. Gray cells denote the best values per column. Train sets each contain only 5 objects; the "All" group contains all 15. The d_C column measures the best Chamfer distance between the test geometry and the train geometries. Lower d_C implies geometric similarity between the train and test objects, and corresponds to more favorable MAE and τ during prediction. Additional metrics and visualizations are on our website: `https://sites.google.com/view/defgraspnets`.

Train set	Mustard bottle					Lemon half					Strawberry				
	d_C [mm] ↓	Deformation [mm]		Stress [kPa]		d_C ↓	Deformation		Stress		d_C ↓	Deformation		Stress	
		MAE ↓	τ_d ↑	MAE ↓	τ_s ↑		MAE ↓	τ_d ↑	MAE ↓	τ_s ↑		MAE ↓	τ_d ↑	MAE ↓	τ_s ↑
Group 1	5.57	0.71	0.62	2.92	0.56	3.57	4.82	0.20	2.15	0.31	3.27	0.42	0.09	6.41	0.58
Group 2	6.77	0.74	0.20	4.72	0.45	3.30	3.98	0.50	1.30	0.43	2.61	0.30	0.29	2.85	0.54
Group 3	6.07	0.73	-0.31	4.57	0.45	4.50	4.28	-0.03	2.63	0.19	2.46	0.31	-0.05	2.79	0.64
All	5.57	0.73	0.60	3.66	0.56	3.30	3.98	0.43	1.36	0.43	2.46	0.30	0.39	2.68	0.66

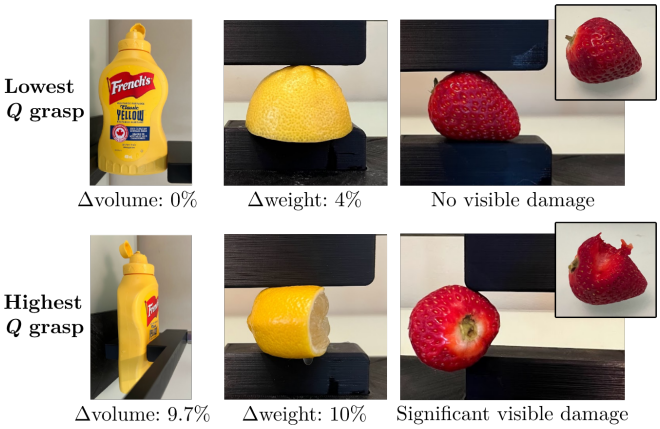


Fig. 6: Validation of grasps from Fig. 5 using a Franka-based gripper gravitationally loaded under 15N. For the bottle and lemon, deformation is measured by proxy (change in volume and weight). For the strawberry, only the highest-Q grasp imparts damage.

metric for mean $\vec{d}$ and $\vec{\sigma}$. We address the following questions:

- Does jointly predicting stress and deformation outperform using two separate networks to predict these quantities? (Ans: The τ metric is comparable in both cases, likely because stress and deformation are coupled through the equations of elasticity. Thus, training two networks would be strictly disadvantageous computationally, c.f. V1.)
- Does one-step prediction outperform multi-step prediction? (Ans: Yes, when predicting deformation. Otherwise, both are comparable when predicting stress, c.f. V2. In MeshGraphNets, multi-step prediction does not accumulate significant deformation errors because the trajectory of the actuators is exactly controlled. In DefGraspNets, gripper force is commanded; the positions of *both* M_o and M_g are predicted and subject to accumulating errors.)
- Should F_g be normalized by the number of contact edges? (Ans: Yes. This aligns with simulation, in which the total force is the sum of forces at all contact points, c.f. V3.)
- Should force features F_g^C be assigned to contact edges or to gripper nodes? (Ans: The network is able to incorporate this information equally well, c.f. V4.)

IX. Discussion and Future Work

We present DefGraspNets, a differentiable GNN-based model for FEM simulation of 3D stress and deformation fields. We demonstrate that training DefGraspNets on a diverse set of grasps on primitive geometries enables effective prediction and grasp planning on unseen, real-world geometries. DefGraspNets enables not only fast evaluation of sampled candidate grasps (1500x faster than GPU-accelerated FEM), but also gradient-based refinement of these grasps to optimize field quantities (e.g., max stress and mean deformation). We verify the effectiveness of optimized grasps on novel objects both in the ground-truth FEM simulator and in the real world.

TABLE II: Ablation study variables and conditions. Our DefGraspNets network conditions are in bold text. Best conditions are in gray.

Variable	Condition	τ_d↑	τ_s↑
V1. Num. outputs	**Def. and stress**	0.61	0.82
	Def. only	0.57	
	Stress only		0.84
V2. Prediction type	**One-step predictions**	0.61	0.82
	Multi-step	0.37	0.70
V3. Value of F_g^W	**Distributed, $\mathbf{F_g/\lvert E^W \rvert}$**	0.61	0.82
	Non-distributed F_g	0.33	0.51
V4. Assignment of F_g	**On world edges E^W**	0.61	0.82
	On all nodes V	0.58	0.74

To expand DefGraspNets for use in downstream manipulation tasks such as food preparation or robotic surgery, prediction of additional quantities should be explored. These may include stability during transport and deformation and flow under reorientation and gravity. Furthermore, as FEM simulators evolve, DefGraspNets can be retrained to predict soft-soft contact or heterogeneous material responses.

Developing data augmentation techniques for *meshes* may enable vast dataset scaling from a minimal set of object models, further strengthening our ability to generalize to unseen objects. In addition, as our network is differentiable, techniques such as Stein variational gradient descent [68] and stochastic gradient Langevin dynamics [69] may allow us to provide probabilistic, multi-modal *distributions* of optimal grasps. Finally, architecture optimization (e.g., sparsity acceleration [70]) may lead to even faster performance.

DefGraspNets contributes the first differentiable approach to deformable grasp planning capable of predicting and optimizing stress and deformation fields on novel objects. We believe this coupling of fast prediction of field quantities with a differentiable model will enable a wide range of users to apply deformable grasp planning to their target domains.

X. Acknowledgment

We thank Miles Macklin and Eric Heiden for simulation expertise; Ankur Handa for network design advice; and Balakumar Sundaralingam and Clemens Eppner for insightful discussions.

REFERENCES

[1] M. C. Gemici and A. Saxena. Learning haptic representation for manipulating deformable food objects. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2014.

[2] J. Smolen and A. Patriciu. Deformation planning for robotic soft tissue manipulation. In *Intl. Conf. on Advances in Computer-Human Interactions*, 2009.

[3] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey. *Intl. Journal of Robotics Research*, 2018.

[4] Jihong Zhu, Andrea Cherubini, Claire Dune, David Navarro-Alarcon, Farshid Alambeigi, Dmitry Berenson, Fanny Ficuciello, Kensuke Harada, Jens Kober, Xiang Li, Jia Pan, Wenzhen Yuan, and Michael Gienger. Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 2022.

[5] S. Timoshenko and J.N. Goodier. *Theory Of Elasticity*. McGraw-Hill Education, 2010.

[6] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[7] Matthew T Mason. *Mechanics of Robotic Manipulation*. MIT press, 2001.

[8] Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 2021.

[9] Veronica E. Arriola-Rios, Puren Guler, Fanny Ficuciello, Danica Kragic, Bruno Siciliano, and Jeremy L. Wyatt. Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, 2020.

[10] Isabella Huang, Yashraj Narang, Clemens Eppner, Balakumar Sundaralingam, Miles Macklin, Ruzena Bajcsy, Tucker Hermans, and Dieter Fox. DefGraspSim: Physics-based simulation of grasp outcomes for 3D deformable objects. *IEEE Robotics and Automation Letters*, 2022.

[11] Yashraj S Narang, Balakumar Sundaralingam, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. Interpreting and predicting tactile signals for the syntouch biotac. *Intl. Journal of Robotics Research*, 2021.

[12] Yashraj Narang, Balakumar Sundaralingam, Miles Macklin, Arsalan Mousavian, and Dieter Fox. Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections. In *IEEE Intl. Conf. on Robotics and Automation*, 2021.

[13] Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. DiSECt: A differentiable simulator for parameter inference and control in robotic cutting. *Autonomous Robots*, 2022.

[14] Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. Accelerated policy learning with parallel differentiable simulation. In *Intl. Conf. on Learning Representations*, 2022.

[15] Roderic A Grupen. Planning grasp strategies for multifingered robot hands. In *IEEE Intl. Conf. on Robotics and Automation*, 1991.

[16] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 2012.

[17] M. Ciocarlie, Corey Goldfeder, and P. Allen. Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2007.

[18] Rhys Newbury, Morris Gu, Lachlan Chumbley, Arsalan Mousavian, Clemens Eppner, Jürgen Leitner, Jeannette Bohg, Antonio Morales, Tamim Asfour, Danica Kragic, Dieter Fox, and Akansel Cosgun. Deep learning approaches to grasp synthesis: A review, 2022.

[19] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *Intl. Journal of Robotics Research*, 2015.

[20] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DOF GraspNet: Variational grasp generation for object manipulation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[21] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken 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.

[22] Qingkai Lu, Mark Van der Merwe, Balakumar Sundaralingam, and Tucker Hermans. Multi-fingered grasp planning via inference in deep neural networks. *IEEE Robotics & Automation Magazine*, 2020.

[23] Jens Lundell, Enric Corona, Tran Nguyen Le, Francesco Verdoja, Philippe Weinzaepfel, Grégory Rogez, Francesc Moreno-Noguer, and Ville Kyrki. Multi-FinGAN: Generative coarse-to-fine sampling of multi-finger grasps. In *IEEE Intl. Conf. on Robotics and Automation*, 2021.

[24] K. Gopalakrishnan and K. Goldberg. D-space and deform closure grasps of deformable parts. *Intl. Journal of Robotics Research*, 2005.

[25] Yan-Bin Jia, Feng Guo, and Huan Lin. Grasping deformable planar objects: Squeeze, stick/slip analysis, and energy-based optimalities. *Intl. Journal of Robotics Research*, 2014.

[26] Peng Song, Juan Antonio Corrales Ramón, and Youcef Mezouar. Dynamic evaluation of deformable object grasping. *IEEE Robotics and Automation Letters*, 2022.

[27] Tran Nguyen Le, Jens Lundell, Fares J. Abu-Dakka, and Ville Kyrki. A novel simulation-based quality metric for evaluating grasps on 3D deformable objects. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2023.

[28] J. Xu, M. Danielczuk, J. Ichnowski, J. Mahler, E. Steinbach, and K. Goldberg. Minimal work: A grasp quality metric for deformable hollow objects. In *IEEE Intl. Conf. on Robotics and Automation*, 2020.

[29] Koshi Makihara, Yukiyasu Domae, Ixchel G. Ramirez-Alpizar, Toshio Ueshiba, and Kensuke Harada. Grasp pose detection for deformable daily items by pix2stiffness estimation. *Advanced Robotics*, 2022.

[30] Z. Pan, X. Gao, and D. Manocha. Grasping fragile objects using a stress-minimization metric. In *IEEE Intl. Conf. on Robotics and Automation*, 2020.

[31] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *Intl. Conf. on Learning Representations*, 2019.

[32] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020.

[33] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *Intl. Conf. on Learning Representations*, 2020.

[34] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *Intl. Conf. on Learning Representations*, 2021.

[35] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. In *Robotics Science and Systems*, 2022.

[36] Timothy P Lillicrap and Adam Santoro. Backpropagation through time and the brain. *Current Opinion in Neurobiology*, 55:82–89, 2019.

[37] Nicolas Lauzier. Robot force control: An introduction, 2016. `https://blog.robotiq.com/bid/53553/Robot-Force-Control-An-Introduction`.

[38] OnRobot. The power and importance of sensing technologies, 2019. `https://onrobot.com/en/blog/the-power-and-importance-of-sensing-technologies`.

[39] Schunk. Parallel gripper: From micro assembly to heavy-load handling, 2021. `https://schunk.com/ca_en/gripping-systems/category/gripping-systems/schunk-grippers/parallel-gripper/`.

[40] Miles Macklin. Warp: A high-performance Python framework for GPU simulation and graphics. `https://github.com/nvidia/warp`, March 2022. NVIDIA GPU Technology Conference (GTC).

[41] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax: A differentiable physics engine for large scale rigid body simulation. `http://github.com/google/brax`, 2021.

[42] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. In *Intl. Conf. on Learning Representations*, 2020.

[43] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. NeuralSim: Augmenting differentiable simulators with neural networks. In *IEEE Intl. Conf. on Robotics and Automation*, 2021.

[44] Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C Karen Liu. Fast and feature-complete differentiable physics for articulated rigid bodies with contact. In *Robotics Science and Systems*, 2021.

[45] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. In *Intl. Conf. on Learning Representations*, 2020.

[46] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. ChainQueen: A real-time differentiable physical simulator for soft robotics. In *IEEE Intl. Conf. on Robotics and Automation*, 2019.

[47] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. PlasticineLab: A soft-body manipulation benchmark with differentiable physics. In *Intl. Conf. on Learning Representations*, 2021.

[48] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. ADD: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 2020.

[49] Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An end-to-end differentiable framework for contact-aware robot design. In *Robotics Science and Systems*, 2021.

[50] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 2018.

[51] Charles C Margossian. A review of automatic differentiation and its efficient implementation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2019.

[52] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 2019.

[53] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 2021.

[54] Samuel Pfrommer, Mathew Halm, and Michael Posa. ContactNets: Learning discontinuous contact dynamics with smooth, implicit representations. In *Conference on Robot Learning*, 2020.

[55] Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[56] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. DiffPD: Differentiable projective dynamics. *ACM Transactions on Graphics (TOG)*, 2021.

[57] Yiling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. Differentiable simulation of soft multi-body systems. In *Advances in Neural Information Processing Systems*, 2021.

[58] Peter C Horak and Jeff C Trinkle. On the similarities and differences among contact models in robot simulation. *IEEE Robotics and Automation Letters*, 2019.

[59] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[60] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Viktor Makoviychuk. Non-smooth Newton methods for deformable multi-body dynamics. *ACM Transactions on Graphics (TOG)*, 2019.

[61] Prajjwal Jamdagni and Yan-Bin Jia. Real food dataset, 2021.

[62] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. *ACM Trans. on Graphics*, 2020.

[63] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. A billion ways to grasp: An evaluation of grasp sampling schemes on a dense, physics-based grasp data set. In *Int. Symp. on Robotics Research*, 2019.

[64] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, USA, 2006.

[65] P. J. M. Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, USA, 1987.

[66] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.

[67] TurboSquid. 3d models for professionals. `https://www.turbosquid.com`, 2023.

[68] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[69] Max Welling and Yee Whye Teh. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the International Conference on Machine Learning*, pages 681–688. Omnipress, 2011.

[70] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.