Grasp'D: Differentiable Contact-rich Grasp Synthesis for Multi-fingered Hands Supplementary Material

Dylan Turpin[1,2,3], Liquan Wang[1,2], Eric Heiden[3], Yun-Chun Chen[1,2], Miles Macklin[3], Stavros Tsogkas[4], Sven Dickinson[1,2,4], and Animesh Garg[1,2,3]

[1] University of Toronto, [2] Vector Institute, [3] Nvidia, [4] Samsung
dylanturpin@cs.toronto.edu

Overview

In this supplementary material, we provide additional details and results to complement the main paper. Specifically:

-- We describe the details of our implementation and experimental setting. (Appendix A).
-- We provide additional results of our method applied to the YCB dataset [2] with both a human MANO hand model [14] and a robotic Allegro hand model. (Appendix B).
-- We provide visualizations of optimization trajectories for MANO hand grasps of YCB and ShapeNet objects, which show how grasps improve as optimization progresses. (Appendix C)
-- We provide additional results for the validation of grasp synthesis with RGB-D reconstruction presented (Section 4.3 of the main paper). (Appendix D).

A Details of implementation and experiments

A.1 Dataset listings

-- Table 1 - ShapeNet object listing (for main experiment in section 4.2).
-- Table 2 - YCB object listing (for rgb-d experiment in section 4.3).
-- Table 3 - YCB object listing (for ablation experiment in section 4.4).

A.2 Initialization and smoothing schedule

Initialization. Since our grasp synthesis pipeline relies on gradient-based optimization, the final result depends on how the parameters are initialized, i.e., different initial hand poses will recover different final grasps. This is a useful quality in that it allows us to sample a variety of grasps for each object by sampling different starting poses. The force variables $\widehat{\mathbf{f}}_c$ are always initialized to zero. We employ a simple heuristic (adapted from [1]) to initialize the hand pose $\mathbf{q}_h^{(0)}$. We set all hand joints to their fully open position. To find an initial rotation and position for the hand base link, we uniformly sample an approach point a on the object surface and a roll angle θ around the approach vector. We use an approach vector opposing the object surface normal at the approach point, and set the hand rotation such that the palm's normal is aligned with

the approach vector. Finally we apply the sampled roll θ around the approach vector. We set the hand position so that the palm's center is at a 10cm distance from the approach point along the approach vector.

Coarse-to-fine smoothing schedule. We set the initial value for the coarse-to-fine smoothing radius r to the distance between the object and the closest point on the hand less 1cm. The radius is then decreased to 0 on a linear schedule over the first 5,000 steps of a 7,000 step optimization and remains at 0 for the final 2,000 steps. The early steps of the optimization find a rough pose for the hand (where on the object to grasp and an approximate finger configuration) and the later steps optimize over fine-grained geometry, allowing the discovery of grasps that conform closely to detailed surface geometry.

A.3 Mesh processing

We use a discretized SDF representation as described in Section 3.2 of the main paper. Computing the SDF involves some preprocessing. For the experiments in Section 4.2 and 4.4 (on complete ShapeNet and YCB meshes respectively), the input is a mesh from the relevant dataset. For the RGB-D reconstruction experiment in Section 4.3, the input is a reconstructed mesh (see Appendix A.4 for details of the reconstruction pipeline). To compute the sign of the SDF at a given query point, we must determine whether that point is inside or outside the object. This is more straightforward if the mesh consists of a single closed surface, so we first run ManifoldPlus [8] to compute a watertight mesh. Next, a $(256 \times 256 \times 256)$ grid of points is evenly sampled over the mesh bounding box (padded by 1cm) and the signed distance of each point to the mesh is computed using libigl [9].

A.4 Reconstruction pipeline

We describe the RGB-D reconstruction pipeline used in Section 4.3. The YCB object dataset includes RGB-D captures for each object. The object is placed on a spinning platter surrounded by 5 cameras and is captured at each of 120 different angles as the plate is rotated in 3 degree increments. We take 15 of these depth images (captures from the first, third and fifth camera at 5 angles in 72 degree increments). We run the code provided alongside the YCB dataset in order to register the depth maps and combine them into a single world frame point cloud. We create a Poisson reconstruction [11] of this point cloud using the Open3D library [17] with a depth of 5. The resulting mesh is still incomplete because the bottom of the object is not visible (since it is the contact surface between the table and the object) We use PyMeshFix [15] to complete this and any other remaining holes in the mesh.

A.5 Simulation details

We run each simulation for a single timestep of length 1×10^{-5} seconds. For the MANO hand model, all 773 vertices are used as contact locations. For the Allegro hand model, we sample $\sim$3000 points on the surface to use as contact locations. In all experiments we set the normal stiffness to $k_n = 1 \times 10^6$, frictional stiffness to $k_f = 1 \times 10^8$, and the friction coefficient to $\mu = 0.8$. For the leaky gradient (described in Section 3.3 of the main paper) we set the proportion of

gradient that leaks through non-colliding contact forces to $\alpha = 0.1$. Note that the above applies to simulation during grasp optimization. When we compute simulation displacement for evaluation purposes, we do not use our own simulator, but instead use PyBullet [5] (details in Appendix A.6).

A.6 Evaluation details

We evaluate grasps in terms of their contact patterns and stability.

Interpenetration volume is the volume of the intersection between the hand and the object. Lower values are better (since in reality the hand cannot penetrate the hard object). We compute this by voxelizing the hand (with 1mm resolution) and querying the object's SDF at each voxel position to decide if each hand voxel is overlapping the object or not.

Contact area is the area of surface contact (in cm^2) between the hand and the object. This is computed similarly to interpenetration volume, except that only the hand surface is voxelized (i.e., the hand is treated as an empty shell, not a solid volume).

Contact area to interpenetration volume ratio. Interpenetration can be avoided by simply avoiding contact with the object entirely, so there is a trade off between interpenetration volume and the other metrics. To capture the amount of interpenetration, conditional on the amount of contact, we report the ratio of contact area to interpenetration.

ϵ (Ferrari-Canny) metric measures grasp stability using the magnitude of the smallest force that can break a grasp. A more stable grasp can withstand larger forces, so a larger force will be needed to break the grasp. This quantity is equivalent to the size of the largest origin-centered ball contained in the Grasp Wrench Space (GWS [6]). The GWS is the space of wrenches the contacts induced by the grasp can withstand, assuming that the total hand-object wrench will be a linear combination of the wrenches at each contact with coefficients summing to 1. Under a Coulomb friction model, the possible wrenches at each contact are defined by a friction cone, which we approximate by a pyramid.

Volume metric is an alternate measure of stability that considers all the possible forces a grasp can withstand (instead of just the magnitude of the smallest force that breaks the grasp). The volume metric is simply the volume of the GWS.

Simulation displacement is a simulation-based, rather than analytic, measure of stability. We use GANHand's implementation [4] of a simulation displacement metric in PyBullet [5] to measure grasp stability by checking how far the object is displaced from its initial pose when the grasp is applied. We use the default physics parameters provided by GANHand except for setting the friction coefficient to 1.2 (instead of the seemingly high default of 3.0). Whereas we train (i.e., optimize) our grasps in our own custom simulator, this metric is computed in a widely used third-party simulator (PyBullet), with a different collision detector, contact model and time stepping scheme. This avoids giving ourselves an unfair advantage by training (optimizing) and testing (computing evaluation metrics) with the same contact model and physics engine (which baselines we compare to did not have access to).

True signed distances. Whenever a metric relies on the object SDF (e.g., to compute contact forces or to determine if a voxel is intersecting the object or not), we compute that SDF with libigl [9] using the ground truth mesh instead of a discrete grid approximation of the SDF.

Evaluating on top k grasps. For our method, we report metrics for the top 2 and top 5 grasps (ordered by simulation displacement -- details below) for each object. For the ObMan dataset, we report the top 2 and top 5 grasps for each object (ordered by their heuristic measure, described below). The ObMan generation procedure uses the GraspIt! simulator to synthesize grasps by optimizing (with simulated annealing) over an analytic metric. Many grasps for each object are generated by running about 70k steps annealing steps. The top 2 grasps are then selected according to a heuristic measure (see Appendix C.2 of [7]) which encourages palm and phalange contact. This heuristic was explicitly added to compensate for the bias of analytic synthesis towards fingertip-only grasps. To test our own method, we generate 10 grasps for each object, each using 7000 optimization steps. We report these metrics over the top 2 and top 5 grasps with the lowest simulation displacement.

A.7 Optimization details

We used the ADAMax [16] optimizer to update the the hand pose parameters $\mathbf{q}_{\text{h}}^{(0)}$ (with a learning rate of 3×10^{-3}) and force parameters $\widehat{\mathbf{f}}_{\text{c}}$ (with a learning rate of 1×10^{-2}). Some objectives are more important than others, so are treated as constraints to satisfy rather than costs to minimize. Specifically, we use the Modified Differential Multiplier Method [13], treating $\mathcal{L}_{\text{task}} < C_{\text{task}}$ and $\mathcal{L}_{\text{limit}} < C_{\text{limit}}$ as constraints, while minimizing $\mathcal{L}_{\text{phys}}$, $\mathcal{L}_{\text{range}}$ and $\mathcal{L}_{\text{inter}}$. We set $C_{\text{task}} = 1 \times 10^{-4}$ and $C_{\text{limit}} = 1 \times 10^{-4}$. Damping is set to 1.0 for all constraints. During MANO hand experiments, we do not use the joint limit loss $\mathcal{L}_{\text{limit}}$ or joint limit constraint, as these limits appear to be well-handled implicitly by the PCA parameterization. Similarly, we do not compute the self-intersection loss for the MANO hand, yet recover grasps with low self-intersection due to the hand parameterization. All losses are used enabled for the Allegro hand.

A.8 Timing

On a mobile Nvidia RTX 2070, generating a MANO hand grasp for a YCB object (by taking 7,000 optimizer steps) takes about 5 minutes. The MANO hand has only 773 vertices, so the memory footprint of the simulation is limited and three grasps can be synthesized in parallel, reducing average grasp synthesis time to about 2 minutes. While not yet approaching realtime performance, this is comparable to the speed of analytic synthesis with the GraspIt! simulator [12], which takes around 5 minutes [4] to synthesize a grasp when using the eigengrasp planner with simulated annealing as for the ObMan dataset [7].

B YCB results

We provide additional examples of applying our method to objects from the YCB dataset with both the Allegro robotic hand and the MANO human hand models (see Figures 1 and 2 respectively).

C Optimization trajectories

To visualize how grasps evolve as optimization progresses, we render hand poses at regular intervals throughout optimization trajectories for 10 grasps (5 for objects from YCB, 5 for objects from ShapeNet -- see Figure 3 and Figure 4 respectively).

D Additional RGB-D results

We provide additional qualitative results of applying our methods to objects reconstructed from RGBD images in the YCB dataset. Figure 5 shows 3 synthesized grasps for each object visualized from 2 different viewpoints. We also provide quantitative results for our RGB-D experiment (section 4.3 of the main paper) in Table 4.

Input	CA↑	IV↓	$\frac{\text{CA}}{\text{IV}}$↑	SD ↓
GT-Mesh	**42.6**	**2.83**	**15.1**	**0.41**
RGB-D	25.46	7.82	3.26	5.68

Table 4: RGB-D experiment quantitative results. Performance with reconstructions is poorer than with ground truth object models, but the resulting grasps are still visually plausible (see Figure 4 of the main paper, Figure 5 of the supplemental) and metrics are comparable to Grasping Field [10] and GANHand [4]. Small errors in reconstruction may produce large errors in grasp synthesis, a drawback future work might address by optimizing over the reconstruction alongside the grasp.

E Training on synthesized data

Fine-tuning Grasping Field [10] with data generated by Grasp'D improves performance on unseen YCB objects better than additional GraspIt! [12] data. Table 5 displays the result of fine-tuning a pre-trained Grasping Field network with additional data synthesized by either the GraspIt! simulator or Grasp'D. The network is first trained for 1400 epochs on the ObMan dataset [7] (of synthetic GraspIt! grasps) and then fine-tuned for 100 epochs on 1000 new grasps (of ShapeNet [3] objects already included in the ObMan dataset) before final testing on 8 objects from the YCB set. Fine-tuning with Grasp'D data results in significantly higher-contact grasps. This comes with a slight increase in intersection volume, but the ratio of contact area to intersection is improved, as is the simulation displacement.

Data source	CA↑	IV↓	$\frac{\text{CA}}{\text{IV}}$↑	SD ↓
GraspIt! [58]	15.78	**9.44**	1.67	3.30
Grasp'D	**21.00**	11.23	**1.78**	**2.88**

Table 5: Fine-tuning Grasping Field [10] with data generated by Grasp'D improves performance on unseen YCB objects better than additional GraspIt! [12] data.

Fig. 1: Robotic grasping with the four-fingered Allegro hand. Our method works equally well with robotic and human hand models. We visualise grasps of three YCB objects [2] with the four-fingered Allegro robotic hand. We can recover a variety of grasps for each object by sampling different initial hand poses (which gradient-based optimization takes to different final grasps). See Appendix A.2 for details of initialization.

Fig. 2: Synthesized MANO hand grasps of YCB objects. Our method generates contact-rich grasps for objects from the YCB dataset. These qualitative results are drawn from the ablation study in Section 4.4 of the main paper, specifically with all features turned on, corresponding to the row labelled Grasp'D in Table 3.

Category ID	Shape ID
2876657	1071fa4cddb2da2fc8724d5673a063a6
2876657	109d55a137c042f5760315ac3bf2c13e
2876657	10dff3c43200a7a7119862dbccbaa609
2876657	10f709cecfbb8d59c2536abb1e8e5eab
2876657	114509277e76e413c8724d5673a063a6
2876657	1349b2169a97a0ff54e1b6f41fdd78a
2876657	134c723696216addedee8d59893c8633
2880940	12ddb18397a816c8948bef6886fb4ac
2880940	13e879cb517784a63a4b07a265cff347
2880940	154ab09c67b9d04fb4971a63df4b1d36
2880940	18529eba21e4be8b5cc4957a8e7226be
2880940	188281000adddc9977981b941eb4f5d1
2880940	1a0a2715462499fbf9029695a3277412
2880940	1b4d7803a3298f8477bdcb8816a3fac9
2942699	1298634053ad50d36d07c55cf995503e
2942699	147183af1ba4e97b8a94168388287ad5
2942699	15e72ce7a8a328d1fd9cfa6c7f5305bc
2942699	17a010f0ade4d1fd83a3e53900c6cbba
2942699	1967344f80da29618d342172201b8d8c
2942699	1ab3abb5c090d9b68e940c4e64a94e1e
2942699	1cc93f96ad5e16a85d3f270c1c35f1c7
2946921	100c5aee62f1c9b9f54f8416555967
2946921	10c9a321485711a88051229d056d81db
2946921	11c785813efc4b8630eaaf40a8a562c1
2946921	129880fda38f3f2ba1ab68e159bfb347
2946921	147901ede668deb7d8d848cc867b0bc8
2946921	17ef524ca4e382dd9d2ad28276314523
2946921	19fa6044dd31aa8e9487fa707cec1558
2992529	1101db09207b39c244f01fc4278d10c1
2992529	1105c21040f11b4aec5c418afd946fad
2992529	112cdf6f3466e35fa36266c295c27a25
2992529	113303df7880cd71226bc3b9ce9ff2a1
2992529	11e925e3ea180b583388c2584b2f0f90
2992529	11f7613cae7d973fd7e59c29eb25f02f
2992529	128bb46234d7250721844676433a0aca
3593526	10af6bdfd126209faaf0ad030fc37d94
3593526	1168c9e9db2c1c5066639e628d6519b6
3593526	117843347cde5b502b18a5129db1b7d0
3593526	1252b0fc818969ebca2ed12df13a916a
3593526	12d643221a3edaa4ab361b6be63163da
3593526	12ec19e85b31e274725f67267e31c89
3593526	133dc38c1316d9515dc3653f8341633a
3624134	102982a2159226c2cc34b900bb2492e
3624134	118141f7d22bc46eaeb7b7328341827a
3624134	11c987c9a34457e48c2fa4fb6bd3e62
3624134	135f75a374a1e22c46cb8dd27ae7fcd
3624134	13bf5728b1f3b6cfadd1691b2083e9e7
3624134	13d183a44f143ca8c842482418ab083d
3624134	1460eded8006b10139c78a1e40e247f3
4074963	1941c37c6db30e481ef53acb6e05e27a
4074963	1aa78ce410bdbcd92530f02db7e9157e
4074963	2053bdd83749adcc1e5c09d9fe5c0c76
4074963	226078581cd4efd755c5278938766a05
4074963	240456647fbca47396d8609ec76a915b
4074963	25182f6e03375c9e7b6fd5468f603b31
4074963	259539bd48513e3410d32c800df6e3dd

Table 1: For experiment 1 (comparison to Obman) in Section 4.2 of the main paper, we use the following ShapeNet objects.

Object ID	Object Name
001	chips_can
002	master_chef_can
005	tomato_soup_can
006	mustard_bottle
008	pudding_box
010	potted_meat_can
021	bleach_cleanser
035	power_drill

Table 2: For experiment 2 (RGBD reconstruction) we use the following YCB objects.

Object ID	Object Name
002	master_chef_can
003	cracker_box
004	sugar_box
005	tomato_soup_can
006	mustard_bottle
007	tuna_fish_can
008	pudding_box
009	gelatin_box
010	potted_meat_can
011	banana
019	pitcher_base
021	bleach_cleanser
024	bowl
025	mug
035	power_drill
036	wood_block
037	scissors
040	large_marker
051	large_clamp
052	extra_large_clamp
061	foam_brick

Table 3: For experiment 2 (RGBD reconstruction) we use the following YCB objects.

Fig. 3: Optimization trajectories for MANO hand grasps of YCB objects. Grasps improve as optimization progresses (from left-to-right in the figure). We visualize the optimization paths that result in the final grasps in Figure 2. The leftmost column shows the initial hand pose (see Appendix A.2 for details of initialization) and the optimization progresses from left to right until reaching the final grasps in the rightmost column. Initially, the hand is not even in contact with the object, but as optimization continues the grasp becomes higher contact, more plausible, and more stable.

Fig. 4: Optimization trajectories for MANO hand grasps of ShapeNet objects. Grasps improve as optimization progresses (from left-to-right in the figure). We visualize the optimization paths that result in the final grasps in the second row of Figure 1 of the main paper.

Fig. 5: Grasp synthesis from RGB-D. We use RGB-D captures from the YCB dataset [2] to reconstruct object models from which we synthesize grasps (see section 4.3 of the main paper for details). Our method can synthesize plausible grasps not just from ground truth object models, but also from imperfect reconstructions.

Bibliography

[1] Brahmbhatt, S., Handa, A., Hays, J., Fox, D.: Contactgrasp: Functional multi-finger grasp synthesis from contact. In: IROS (2019) 1

[2] Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., Dollar, A.M.: Yale-cmu-berkeley dataset for robotic manipulation research. International Journal of Robotics Research (2017) 1, 7, 12

[3] Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) 5

[4] Corona, E., Pumarola, A., Alenya, G., Moreno-Noguer, F., Rogez, G.: Ganhand: Predicting human grasp affordances in multi-object scenes. In: CVPR (2020) 3, 4, 5

[5] Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org (2016) 3

[6] Ferrari, C., Canny, J.F.: Planning optimal grasps. In: ICRA (1992) 3

[7] Hasson, Y., Varol, G., Tzionas, D., Kalevatykh, I., Black, M.J., Laptev, I., Schmid, C.: Learning joint reconstruction of hands and manipulated objects. In: CVPR (2019) 4, 5

[8] Huang, J., Zhou, Y., Guibas, L.: Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. arXiv preprint arXiv:2005.11621 (2020) 2

[9] Jacobson, A., Panozzo, D., et al.: libigl: A simple C++ geometry processing library (2018) 2, 4

[10] Karunratanakul, K., Yang, J., Zhang, Y., Black, M.J., Muandet, K., Tang, S.: Grasping field: Learning implicit representations for human grasps. In: 3DV (2020) 5, 6

[11] Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Eurographics symposium on Geometry processing. vol. 7 (2006) 2

[12] Miller, A.T., Allen, P.K.: Graspit! a versatile simulator for robotic grasping. IEEE Robotics & Automation Magazine (2004) 4, 5, 6

[13] Platt, J.C., Barr, A.H.: Constrained differential optimization. In: NeurIPS (1987) 4

[14] Romero, J., Tzionas, D., Black, M.J.: Embodied hands: Modeling and capturing hands and bodies together. ACM Transactions on Graphics (2017) 1

[15] Sullivan, C.B., Kaszynski, A.: PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). Journal of Open Source Software (2019) 2

[16] Zhang, Z.: Improved adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). pp. 1--2. IEEE (2018) 4

[17] Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018) 2