Abstract—Teaching a multi-fingered dexterous robot to grasp objects in the real world has been a challenging problem due to its high dimensional state and action space. We propose a robot-learning system that can take a small number of human demonstrations and learn to grasp unseen object poses given partially occluded observations. Our system leverages a small motion capture dataset and generates a large dataset with diverse and successful trajectories for a multi-fingered robot gripper. By adding domain randomization, we show that our dataset provides robust grasping trajectories that can be transferred to a policy learner. We train a dexterous grasping policy that takes the point clouds of the object as input and predicts continuous actions to grasp objects from different initial robot states. We evaluate the effectiveness of our system on a 22-DoF floating Allegro Hand in simulation and a 23-DoF Allegro robot hand with a KUKA arm in real world. The policy learned from our dataset can generalize well on unseen object poses in both simulation and the real world.

I. INTRODUCTION

Our world is largely designed for human hands, anthropomorphic robotic hands are likely to be an important element of robotic systems in human-centric environments. To build general purpose dexterous manipulation systems, we require methods that are able to train policies that can generalize widely with relatively little burden placed on a human supervisor. In this work, our goal is to build a system that is able to leverage a small amount of human supervision to learn a robust controller for dexterous manipulation tasks, which can be reliably deployed on a real robot.

We proposed a demonstration guided data augmentation system (Fig.2) that aims to generate a large dataset of diverse, successful trajectories for a robot gripper in simulation. Our data-augmentation pipeline combines motion retargeting with local gradient-free trajectory refinement and augmentation to obtain a large variety of successful data relatively cheaply. This data can be used to learn policies mapping from point-clouds to actions, which can be transferred to real world scenarios to grasp objects in novel poses.

While human demonstration plays a crucial role in training manipulation policies, devising a scalable method to collect demonstrations is still a key bottleneck. Prior work has collected demonstrations kinesthetically [45], through VR interfaces [35], or using motion capture (mocap) solutions [9, 15, 40, 33]. Some recent efforts have also explored collecting demonstrations by harvesting internet video [23]. Although there has been some innovations for parallel-jaw grippers [19, 38, 43], acquiring demonstrations for multi-fingered dexterous manipulators remains challenging. Consequently, the collected demonstration datasets are often limited in size. Our work leverages existing mocap datasets of human grasping [4] to guide the generation of a large, diverse grasping dataset for
training dexterous manipulators.

II. DexTransfer: A System for Real World Dexterous Grasping with Minimal Human Demonstrations

Fig. 2. Overview of the proposed framework DexTransfer. The mocap data is first retargeted to a dexterous gripper in simulation. This motion reference is then refined and augmented into a large and diverse set of successful trajectories, to learn a policy to succeed on unseen object poses and initial hand poses. The learned policy is eventually transferred to a real robot system.

Our work aims to build a system that can leverage a small number of human demonstrations to learn grasping policies for dexterous grippers which are robust across a variety of object poses in the real world. To accomplish this, our system consists of three major phases - (1) trajectory retargeting (2) trajectory refinement and (3) policy learning, that are performed in sequence to acquire dexterous grasping policies as illustrated in Figure 2. The system starts by being provided with human demonstrations, which are first retargeted to an actual robot gripper in simulation as described in Section II-A. Since the retargeted trajectories are not guaranteed to be dynamically successful, the system then aims to generate trajectory refinements that leads to robust and successful grasps as described in Section II-B along with performing directed trajectory augmentation to increase the dataset size and diversity. Lastly, these successful grasp trajectories can be used for policy learning directly from sensory observations in Section II-C.

A. Trajectory Retargeting

Our system takes a set of human demonstrations provided via a hand motion capture system. Precisely, we assume a demonstration dataset $\mathcal{D}_{\text{human}} = \{\tau_0^h, \tau_1^h, \ldots, \tau_N^h\}$, where each demonstration $\tau_i = \{(q_0^h, o_0^h), (q_1^h, o_1^h), \ldots, (q_{T_{\tau}}^h, o_{T_{\tau}}^h)\}$ is a trajectory of hand 3D pose $q_t$ and object 3D pose $o_t$. These demonstrations, when directly mapped to a robot hand using standard Inverse Kinematics (IK), are typically not functional due to the mismatch between the human and robot actuator shape and kinematics.

To address this issue, we follow DexPilot [13] and formulate the retargeting objective as a non-linear optimization problem with a general cost function that is able to retarget human data to various robot hands of different morphologies while preserving the original demonstration behaviors.

$$d_f = \sum_{i=0}^{N} ||r_i(q_{\tau_f}) - s_r(\hat{r}_i(q_{\tau_h}))||^2,$$

$$d_{\text{obj}} = ||\alpha(\tau_f) - \alpha(\hat{\tau}_i)||_{2},$$

$$d_r = G(M_h, M_r),$$

$$\arg\min_{(q_{\tau_f})} (w_f d_f + w_{\text{obj}} d_{\text{obj}} + w_r d_r),$$

Eq. 1 is the same as the cost function defined in [13], where each $r_i(q_{\tau_f})$ and $\hat{r}_i(q_{\tau_h})$ are displacement vectors between hand joints for robot and human hands, and $s_r$ is the scale ratio between allegro and a human hand. To move the robot hand towards the object in a similar way as the demonstration, we introduce Eq. 2 where $\alpha(\tau_f)$ and $\alpha(\hat{\tau}_i)$ represents displacements between the object center and each finger tip. We also add Eq. 3 where $G$ is the minimum geodesic distance in SO(3) between the orientation of the human palm $M_h$ and robot palm $M_r$.

The output of retargeting is a set of demonstrations represented by the joint positions of the robot $\mathcal{D}_{\text{retargeted}} = \{\tau_0^r, \tau_1^r, \ldots, \tau_N^r\}$, with each trajectory $\tau_i^r = \{(h_0^r, e_0^r, o_0^r), (h_1^r, e_1^r, o_1^r), \ldots, (h_{T_{\tau}}^r, e_{T_{\tau}}^r, o_{T_{\tau}}^r)\}$ consisting of finger positions $h_t^r$, end effector positions $e_t^r$ and object poses/pointclouds $o_t^r$. These roughly capture human trajectories while being consistent with the robot kinematics. We parameterize the action space as target joint positions and we run an underlying PID controller, making it trivial to define actions $o_t^r$ from these demonstrations simply as the next position in the trajectory $(h_t^r, e_t^r, o_t^r)$.

B. Trajectory Refinement

Retargeted trajectories simply try to match kinematic poses without actually considering the dynamics of the world, not accounting for unseen contact forces. For contact rich manipulation tasks like dexterous grasping, this can lead to catastrophic failures in trajectory execution. The process of trajectory refinement takes the (potentially unsuccessful) retargeted trajectories $\mathcal{D}_{\text{retargeted}} = \{\tau_0^r, \tau_1^r, \ldots, \tau_N^r\}$ from Section II-A and refines them to a diverse set of successful trajectories on a variety of different objects. The process of trajectory refinement consists of (1) Generating a large set of nominal trajectories to refine from retargeted trajectories via template matching (2) perturbing and refining these nominal trajectories to be dynamically successful at task completion (3) augmenting refined trajectories to be diverse across poses, configurations and initial hand states. We describe each of these components in detail.

Template Matching for Nominal Trajectory Generation

Typically we are only given a handful of demonstrations by a human supervisor. For data driven policy learning methods as described in Section II-C this is hardly sufficient and we need to generate a significantly larger dataset to learn general purpose policies. The first insight we leverage is that for rigid
bodies, when we apply a transformation $T$ of the object pose in SE(3) space, the same transformation $T$ can also be applied to the end-effector trajectories $e^i_t$ (in most cases) to yield sensible trajectories $T e^i_t$ for the new object pose as well. For each initial object pose $o^j_0$ that is encountered in a particular demonstration $\tau_j$, we can estimate the rigid transform $T_{o^j_0}$ to the object pose $o^j_0$ in a different demonstration $\tau_j$ and use it to transform the first demonstrations end effector trajectory $e^i_t$ to the second objects frame of reference $T_{o^j_0} e^i_t$ and vice versa. More precisely,

$$o^j_0 = T_{o^j_0} e^i_t, \forall i, j$$

$$\hat{\tau}_j = \{(h^j_0, T_{o^j_0} e^i_0, o^j_0), (h^j_1, T_{o^j_0} e^i_1, o^j_0), \ldots, (h^j_T, T_{o^j_0} e^i_T, o^j_0)\},$$

This transformation provides a drastically larger number of effective trajectories performing in a variety of different object poses.\footnote{In our work, we restrict the transformations to be between just a set of stable canonical poses and perform after the fact rotations to further increase diversity. We compute stable canonical poses using trimesh by sampling and applying correlated perturbations to the nominal controls of the retargeted trajectory. These correlated perturbations are sampled by first sampling a scalar parameter time $t$ from the uniform distribution $t \sim \mathcal{U}[0,1]$ and then using this parameter to choose a per-joint perturbation $(p^i_k) = t * (p^i_{max})_k + (1-t) * (p^i_{min})_k$ that interpolates between minimum and maximum perturbation values per joint $k$, which can then be added to $a_i^0$ to obtained the perturbed action. This sampling scheme allows for correlated perturbation of the fingers, allowing for perturbation to “open” and “close” the fingers more coherently, rather than simply doing uniformly random exploration independently per-joint, which makes refinement significantly less effective.

To generate successfully refined trajectories, we can generate perturbations in the control as described above, simulate trajectories via standard forward simulation and then reject trajectories which don’t meet the desired stability (object does not fall under perturbation and randomization of parameters like mass and friction and is lifted high off the table) and success criteria (object is grasped and lifted). Our choice of rejection condition during rejection sampling can allow us to select for the most robust behaviors as described in Algorithm I-B.

**Object-Centric Augmentation.** While refinement yields dynamically consistent trajectories $D_{\text{refined}}$ that successfully apply contact forces to grasp objects, it still only covers a somewhat narrow diversity of object, hand and arm poses since it performs very local refinements on nominal trajectories. Given that most household objects have multiple plausible grasp positions, which simply differ by a translation offset, we can perform a simple augmentation scheme to propose random translation offsets to refined trajectories and retain those translationally perturbed trajectories that are still able to successfully accomplish the grasping task. This allows us to synthesize a larger dataset of different grasping trajectories beyond the human demonstrations.

While refinement and augmentation do generate a variety of different grasping trajectories, the number of distinct initial positions of the end effector is still somewhat minimal. This
makes policy generalization to novel end effector positions challenging. Given the nature of the grasping problem, we can synthetically generate a much larger dataset \( \mathcal{D}_{\text{augmented}} \) of trajectories with varying initial hand poses by generating a variety of initial hand positions and then doing interpolation in free space to the closest trajectory amongst the existing human provided demonstrations, from where that demonstration can subsequently be executed. We refer this step as data funneling.

Given the combination of template matching, refinement with correlated sampling and object-centric augmentation with funneling, we finally arrive at a diverse and dynamically successful dataset \( \mathcal{D}_{\text{augmented}} \) that is able to successfully apply contact forces and perform grasping from a variety of configurations. This is then used for policy learning as described below.

C. Policy Learning from Point Cloud observations

![Network architecture for the point cloud dependent policy \( \pi_{\theta} \). Scene Encoder consists of three PointNet++ SA modules followed by two fully-connected layers. Kinematics Encoder consists of three residual modules. Fusion Layer takes the concatenated features and feed into one linear layer followed by two residual modules. The network has three branches to predict palm translation, rotation and joint angles. Each branch consists of three residual modules.](image)

Given a robust and diverse dataset of robot trajectories, we train a policy to directly predict the action to execute based on current observation via a standard maximum likelihood supervised learning objective. Our network consists of a **Scene Encoder** and a **Kinematics Encoder**. The Scene Encoder is based on PointNet++ \cite{32}, that takes point clouds of the object as current observation \( O_t \), and predicts scene features as output. The Kinematics Encoder takes the past 5 frames of motions (to deal with partial observability and variable timing) \( f_t, f_{t-1}, ... f_{t-4} \), each includes robot finger joints, keypoints and object center shift, and outputs the kinematics feature. The scene features and kinematics features are then concatenated and fed into Fusion Layer to compute combined features for the current observation. The network splits three branches to predict translation (3-dim), rotations (3-dim) and finger joints (N-dim, N=16 for Allegro). We found that representing observation and action relative to robot palm coordinate is crucial for generalizing to unseen object poses. Thus all the observation and keypoint positions are represented in the current palm coordinates. The predictions of the network are used to compute the loss function as follows, which can then be used to learn policy parameters \( \theta \) via standard stochastic gradient methods:

\[
\min_{\theta} \mathbb{E}_{(h_i^t, e_i^t, o_i^t, a_i^t) \sim \mathcal{D}_{\text{aug}}} \left[ \left| \left| \pi_{\theta}^{\text{translation}}(\cdot \mid h_i^t, e_i^t, o_i^t) - (a_i^t)_{\text{translation}} \right| \right| + G(\pi_{\theta}^{\text{rotations}}(\cdot \mid h_i^t, e_i^t, o_i^t), (a_i^t)_{\text{rotations}}) + \left| \left| \pi_{\theta}^{\text{finger}}(\cdot \mid h_i^t, e_i^t, o_i^t) - (a_i^t)_{\text{finger}} \right| \right| \right]
\]

where \( G \) is the minimum geodesic distance in SO(3).

III. EXPERIMENTS

In this section, we aim to answer the following questions:

1) Can DexTransfer enable us to leverage a small number of human demonstrations to obtain a large number of dynamically successful trajectories in sim across a variety of objects?

2) Can we understand which elements of the proposed system are most crucial to enabling robust dexterous grasping performance?

3) Do the resulting policies transfer to the real world on an Allegro robot hand?

We show videos experiments for both simulation and real world in the supplementary material and further results can be found on the supplementary website [https://sites.google.com/view/dextransfer/home](https://sites.google.com/view/dextransfer/home).

A. Dataset Generation in Simulation

We extracted human demonstrations from the DexYCB dataset \cite{4}, a dataset with 1,000 mocap sequences of humans picking up YCB objects \cite{3} from a table. Each sequence contains a single human subject picking up one specific object with a single hand. The hand pose is captured using the MANO hand model \cite{36}. Given the raw dataset, we curated a subset with 17 objects picked up by a human with the right hand, resulting in a total of 426 demonstration sequences with varying object configurations. Table I reports the accumulated number of trajectories after each processing stage for the Allegro hand in a per object breakdown. Despite MoCap tracking error and kinematics difference between the robot and the human demonstrator, DexTransfer is able to generate more than 10 folds successful data for the allegro robot.

B. Policy Learning on Augmented Dataset

In this section, we show how we can leverage the augmented dataset to learn policies that can perform well at dexterously grasping objects in unseen poses and configurations. In particular, we follow the procedure described in Section II-C and learn a per-object policy that is able to dexterously grasp a particular object in a variety of different poses. During test time, we randomize object 6D poses and initial positions of the robot. We evaluate our policies 20 times on each object.
TABLE I
ALEGGRO HAND DATASET ACROSS 17 YCB OBJECTS. WE TRANSFER A SMALL SET OF MOCAP DATA TO A LARGE DATASET WITH SUCCESSFUL TRAJECTORIES.

| Human Demo | master chef can | cracker box | sugar box | tomato soup can | mustard bottle | tuna fish can |
|------------|----------------|------------|-----------|----------------|---------------|--------------|
| Normalized Trajectory | 26 | 25 | 25 | 25 | 25 | 28 |
| Refined Instance | 78 | 141 | 158 | 56 | 82 | 200 |
| Augmented Trajectory | 182 | 100 | 313 | 146 | 46 | 139 |

| Human Demo | pudding box | gelatin box | potted meat can | wood block | foam brick | bowl |
|------------|-------------|-------------|----------------|------------|-----------|------|
| Normalized Trajectory | 23 | 27 | 23 | 26 | 26 | 27 |
| Refined Trajectory | 87 | 118 | 128 | 102 | 104 | 79 |
| Augmented Trajectory | 40 | 52 | 96 | 55 | 79 | 25 |

| Human Demo | mug | bleach cleaner | power drill | large marker | pitcher base | sum |
|------------|-----|----------------|-------------|--------------|--------------|-----|
| Normalized Trajectory | 23 | 23 | 23 | 26 | 25 | 426 |
| Refined Instance | 101 | 56 | 45 | 43 | 34 | 1621 |
| Augmented Trajectory | 122 | 102 | 123 | 9 | 38 | 2566 |

Fig. 4. Qualitative results of policies on unseen poses of various objects in both simulation and real world. We can see that the hand approaches the objects from a variety of approach angles and grasp poses and can show interesting grasping strategies.

Similar to the set up in [39], each time, if the robot drops the object, we reset our network to additionally captures the rate of success within three attempts. Note that a test episode is ended as soon as a success is reached. The success is defined when the object is above the table by 10 cm. In Table I, we show the success rate of the robot after each attempt. The policy is able to recover and succeed in the next trials because its trained in a closed-loop fashion. All experiments are trained with the same batch size $bs = 460$ and Adam optimizer with learning rate $lr = 0.0002$. We use Nvidia Apex [7] to achieve mixed-precision training through all of our experiments. For depth rendering, we use pyrender [26] to enable fast online rendering during training. We show the policy is able to grasp most objects with a success rate of over 70% on unseen poses and configurations in Table II (DexTransfer (ours)).

C. Understanding Different Elements of the Refinement Pipeline

In this section, we aim to understand which elements of our system are crucial to performance by evaluating the performance of the system as each component: domain randomization, augmentation and data funneling, is removed. In addition, we compare our approach with a heuristic based approach and a nearest neighbor approach. The results are shown in Table II and are described below.

Policy without Domain Randomization We show that adding domain randomization in the dataset can significantly improve policy’s ability to handle covariate shift at inference time. The key insight is one trajectory could be successful by chance, but cannot guarantee the robustness to noise and both across unseen poses and across different objects. We additionally show qualitative results of Allegro hand grasping objects with different poses at different frames in Fig. 4.
different physics parameters. We observe an almost 30% performance drop without domain randomization when evaluating policies on 17 objects as show in the second row in Table II.

**Policy Without Augmentation** We found without augmentation, the policy performs similar on smaller objects but worse on bigger objects like bleach cleaner, power drill and pitcher base. On average, the performance drops more than 10%.

**Policy Without Data Funneling** Without data funneling, the policy fails completely to grasp unseen poses from a novel initial robot state. This is because only refined trajectories cannot cover the entire space. We show that without data funneling, the success rate on average is less than 5%.

**Heuristic Approach** We compare our approach with a heuristic approach, which drives the robot along a linear path to a certain distance from the center of the object pointclouds, and grasp the object using pre-defined finger poses. We observed that this policy only achieves an 11% success rate. This shows that learning to adjust both position and orientation of the robot given the observation of the scene is crucial for success.

**Nearest Neighbor** To illustrate the difference between the training and test set, we further perform the following baseline: For each of the 20 test samples, we find the closest sample in the training set and execute its corresponding actions. We observe only a 14% success rate on average. This suggests our policy learns to adapt to novel poses given the observation of the scene.

**Real Experiments**

Lastly, we test policy transfer to the real world on a physical robot for a subset of the trained policies.

**Robot System** We deploy our policies to a robotic platform that has 23 actuators across a KUKA LBR iiwa 7 R800 robot arm and a Wokin Robotics Allegro robotic hand. We place three RGB-D cameras around the robot to provide necessary point cloud information. Specifically, unseen object instance segmentation [42] is applied to the image data to extract a segmented point cloud of a single object on the table, which is used as input to the trained policies.

**Control** The policies require additional control aspects for real-world application. First, the policy generates palm pose targets, which is an elevated action space for the robot. These pose targets are passed to an underlying, manually-derived geometric fabrics policy that generates high-frequency joint position, velocity, and acceleration targets motion across all joints of the arm. Geometric fabrics is a provably stable,
A second order policy that effectively solves the problem of reaching to end-effector targets while resolving arm redundancy, controlling manipulator posture, avoiding joint limits and joint speed limits, avoiding robot self-collision, and avoiding excessive collision between the robot and the table. The design and tuning of this policy is exactly the same as the one reported in [41].

Finally, the target joint positions generated by fabrics are directly fed to an underlying gravity-compensated joint PD controller at 30 Hz, which are upsampled to 1000 Hz via polynomial interpolation. The trained policies also generate finger joint position commands at 6 Hz, which are fed directly to a PD controller that drives the joints of the Allegro hand. Altogether, the actions of the trained policies are ultimately converted to target joint drive torques that generate motion through the physical robot.

**Evaluation** We tested policies on a subset of objects: Master chef can, potted meat can, mug, cracker box and wood block. In real world, wood block is too heavy for the Allegro hand to lift, so we replace it with a foam block with the same dimensions. To reduce the reflection of the object surface in order to get more reliable segmentation masks, we add non-sticky tapes around cracker box and potted meat can. We test our policies on each object 25 times and report success rate in Figure 5. Qualitative results are shown in Fig. 4. We see that while success rate in the real world is lower than in simulation, it is able to successfully complete the task reliably with over 40% success rate for any object and in many cases closer to 70% success rate.

### IV. CONCLUSIONS

In this work, we proposed a new system for learning to grasp various objects with a multi-fingered dexterous manipulator, leveraging a small number of human-provided demonstrations. Our system retargets the provided demonstrations into simulation, applies a technique for targeted data-augmentation and refinement and then applies large scale supervised learning to learn control policies that operate directly on point-cloud inputs. These policies when trained with sufficient augmentation can be transferred to the real world and can be used to grasp various objects with up to 72% success rate. We show the efficacy of the system in both simulation and the real world, running careful analysis to understand the impact of various design decisions in the system.

There are several directions for future work. For instance, it would be very interesting to go beyond tasks like grasping and do more dexterous problems like in-hand manipulation by continuing to leverage the same general system for simulation driven data augmentation. It would also be interesting to see if we can use the simulator to pretrain policies but continue to finetune them directly in the real world in new domains. Additionally, exploring more complex refinement algorithms like CMA-ES [14] rather than rejection sampling would be a worthwhile exercise.

### REFERENCES

[1] Yunfei Bai and C. Karen Liu. Dexterous manipulation using both palm and fingers. In *ICRA*, 2014.

[2] Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. ContactGrasp: Functional multi-finger grasp synthesis from contact. In *IROS*, 2019.

[3] Berk Calli, Arjun Singh, Aaron Walsman, Siddharth 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)*, 2015.

[4] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. DexYCB: A benchmark for capturing hand grasping of objects. In *CVPR*, 2021.

[5] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *CoRL*, 2021.

[6] Sammy Christen, Muhammed Kocabas, Emre Aksan, Jemin Hwangbo, Jie Song, and Otmar Hilliges. D-Grasp: Physically plausible dynamic grasp synthesis for hand-object interactions. *arXiv preprint arXiv:2112.03028*, 2021.

[7] NVIDIA Corporation. A PyTorch extension: Tools for easy mixed precision and distributed training in PyTorch. https://github.com/NVIDIA/apex, 2020.

[8] Erwin Coumans and Yunfei Bai. PyBullet: a Python module for physics simulation for games, robotics and machine learning. https://pybullet.org, 2016–2021.

[9] Guillermo Garcia-Hernando, Shixin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with RGB-D videos and 3D hand pose annotations. In *CVPR*, 2018.

[10] Guillermo Garcia-Hernando, Edward Johns, and Tae-Kyun Kim. Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning. In *IROS*, 2020.

[11] Michael Gleicher. Retargeting motion to new characters. In *SIGGRAPH*, 1998.

[12] Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *IROS*, 2016.

[13] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. DexPilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *ICRA*, 2020.

[14] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016. URL http://arxiv.org/abs/1604.00772

[15] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *CVPR*, 2019.

[16] Divye Jain, Andrew Li, Shivam Li, Aravind Rajeswaran,
Vikash Kumar, and Emanuel Todorov. Learning deep visuomotor policies for dexterous hand manipulation. In ICRA, 2019.

[17] Rae Jeong, Jost Tobias Springenberg, Jackie Kay, Dan Zheng, Alexandre Galashov, Nicolas Heess, and Francesco Nori. Learning dexterous manipulation from suboptimal experts. In CoRL, 2020.

[18] Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael J. Black, Krikamol Muandet, and Siyu Tang. Grasping field: Learning implicit representations for human grasps. In 3DV, 2020.

[19] Mia Kokie, Danica Kragic, and Jeannette Bohg. Learning task-oriented grasping from human activity datasets. RA-L, 5(2):3352–3359, 2020.

[20] Vikash Kumar, Yuval Tassa, Tom Erze, and Emanuel Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In ICRA, 2014.

[21] Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In ICRA, 2016.

[22] Jeffrey Mahler, Jacky Liang, Sherdi 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 RSS, 2017.

[23] Priyanka Mandikal and Kristen Grauman. DexVIP: Learning dexterous grasping with human hand pose priors from video. In CoRL, 2021.

[24] Priyanka Mandikal and Kristen Grauman. Learning dexterous grasping with object-centric visual affordances. In ICRA, 2021.

[25] Matthew T. Mason and J. Kenneth Salisbury. Robot Hands and the Mechanics of Manipulation. MIT Press, Cambridge, MA, May 1985.

[26] Matthew Matl. Pyrender. https://github.com/mmatl/pyrender, 2019.

[27] Andrew T. Miller and Peter K. Allen. GraspIt! a versatile simulator for robotic grasping. RAM, 11(4):110–122, 2004.

[28] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DOF GraspNet: Variational grasp generation for object manipulation. In ICCV, 2019.

[29] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In CoRL, 2019.

[30] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. arXiv preprint arXiv:1808.00177, 2018.

[31] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik’s cube with a robot hand. arXiv preprint arXiv:1910.07113, 2019.

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++. Deep hierarchical feature learning on point sets in a metric space. In NIPS, 2017.

[33] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. DexMV: Imitation learning for dexterous manipulation from human videos. arXiv preprint arXiv:2108.05877, 2021.

[34] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. In IROS, 2021.

[35] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In RSS, 2018.

[36] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. In SIGGRAPH Asia, 2017.

[37] J. Kenneth Salisbury and John J. Craig. Articulated hands: Force control and kinematic issues. IJRR, 1(1):4–17, 1982.

[38] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6DoF closed-loop grasping from low-cost demonstrations. RA-L, 5(3):4978–4985, 2020.

[39] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 13438–13444. IEEE, 2021.

[40] Omid Taheri, Nima Ghorbani, Michael J. Black, and Dimitrios Tzionas. GRAB: A dataset of whole-body human grasping of objects. In ECCV, 2020.

[41] Karl Van Wyk, Man Xie, Anqi Li, Muhammad Asif Rana, Buck Babich, Bryan Peele, Qian Wan, Iretiayo Akinola, Balakumar Sundaralingam, Dieter Fox, Byron Boots, and Nathan Ratliff. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. RA-L, 2022.

[42] Yu Xiang, Christopher Xie, Arsalan Mousavian, and Dieter Fox. Learning RGB-D feature embeddings for unseen object instance segmentation. In CoRL, 2020.

[43] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. In CoRL, 2020.

[44] He Zhang, Yuting Ye, Takaaki Shiratori, and Taku Komura. ManipNet: Neural manipulation synthesis with a hand-object spatial representation. In SIGGRAPH, 2021.

[45] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In ICRA, 2019.
APPENDIX A
RELATED WORK

a) Manipulation with Dexterous Hands:  Dexterous manipulation has a long history dating back to [37, 25]. More recently, the field has studied the use of dexterous hands in a variety of task domains, ranging from object grasping [24, 23], in-hand manipulation [11, 21, 30, 31], relocation [20, 33], stacking [17], interaction with environmental props [12, 45], tool use [35, 16, 29, 34], to smart teleoperation [13, 10]. Our work addresses the task of object grasping, a key first step to gain hold of an object before any downstream manipulation tasks. Grasping has been conventionally approached by model-based planning [27]. Notably, for simpler end effectors like parallel-jaw grippers, remarkable progress has been recently achieved through end-to-end learning-based approaches which directly predict grasp position from raw visual input, such as a depth image [22] or point cloud [28]. Our work is in line with this direction but addresses the frontier of the more complex dexterous hands.

To control a dexterous hand, prior work has also investigated different approaches, including planning with analytical models [11] and online trajectory optimization [20]. However, these methods assume accurate dynamics models and robust state estimates, which are difficult to obtain in complex real-world manipulation. To overcome this limitation, recent work has resorted to learning-based approaches particularly with deep reinforcement learning (RL). Both model-based RL [21, 12, 29] and model-free RL [30, 31, 17, 5] have been investigated. Despite the progress, training deep RL models remains notoriously challenging due to high sample complexity and tedious reward engineering. Although this issue has been mitigated by incorporating human demonstrations [35, 45, 33, 44, 23], these methods are still faced with a major challenge in scalability—training a single general policy for handling diverse objects and diverse scene configurations is still largely beyond reach [33, 5]. Rather than using deep RL, we adopt the common supervised learning paradigm, where we focus on bootstrapping a large dexterous grasping dataset from a small set of human demonstrations. Our work bears some similarity to [16, 5], where an expert policy is first trained using RL in a privileged state space, followed by training a high-dimensional student policy from expert generated data via behavioral cloning. Instead of training RL experts, our method bootstraps from minimal human demonstrations. Besides, as opposed to many prior works which only evaluate in simulation [55, 16, 17, 24, 34, 23, 5], we evaluate our model on a real-world robot platform.

b) Learning from Human Demonstration:  While human demonstration plays a crucial role in training manipulation policies, devising a scalable means to collect demonstrations is still a key bottleneck. Prior work has collected demonstrations kinesthetically [45], through VR interfaces [35], or using motion capture (mocap) solutions [9, 15, 40, 33]. Although there has been some innovations for parallel-jaw grippers [19, 38, 43], acquiring demonstrations for dexterous manipulators remains challenging. Consequently, the collected demonstration datasets are often limited in size. Our work leverages existing mocap datasets of human grasping [4] to guide the generation of a large, diverse grasping dataset for training dexterous manipulators. Apart from our work, some recent efforts have also explored collecting demonstrations by harvesting internet video [23].

c) Multi-Finger Grasp Synthesis:  Our work connects closely to the task of multi-finger grasp synthesis. Prior work has investigated grasp synthesis for virtual human hands [40, 18] and multi-finger robot manipulators [2]. However, they are only concerned with static grasp poses and do not consider the motion. Some very recent work attempted to bridge this gap by synthesizing trajectories for human-like hands [44, 6]. Our work can potentially capitalize on their grasp trajectories by utilizing them as additional human demonstrations.

APPENDIX B
DEXTRANSFER

A. Generalization on Different Grippers

We further investigate whether DexTransfer is general for other grippers. In addition to Allegro robot, we also generate a dataset for Shadow hand, MANO hand, Robotiq-3f gripper and Kinova. Fig. 6 shows qualitative results on refined poses from human demonstrations. This suggests that DexTransfer is able to transfer human demonstrations to various robot grippers despite the large difference among their kinematic configurations.

B. Details on Trajectory Refinement

Fig. 7 shows details on trajectory refinement and augmentation in DexTransfer. For each nominal trajectory, we sample a scalar parameter time \( t \) from the uniform distribution \( t \sim U[0, 1] \), maximum palm pose \( (p_{\text{max}})^k \) and minimum palm pose \( (p_{\text{min}})^k \). We then use this parameter to interpolate between minimum and maximum perturbation values within the corresponding time steps \( (p_i^k) = t \cdot (p_{\text{max}})^k + (1-t) \cdot (p_{\text{min}})^k \), which can then be added to \( \alpha_i^k \) to obtained the perturbed action. We add domain randomization and only keep refined actions that are successful with 10 different physics parameters (object mass, frictions, noise and perturbation test) via rejection sampling.
Fig. 7. Illustration of trajectory refinement procedure combining refinement, augmentation, and data funneling to get an extended dataset of trajectories for supervised learning. **Left:** Only a few human demonstrations are provided and retargeted to a robot gripper to generate nominal trajectories. **Middle:** grasping trajectories generated by policies on unseen object poses and initial hand poses via refinement and augmentation. **Right:** Extending the set of feasible object poses and novel hand states via data funneling.

**TABLE III**

| EXPERIMENTS ILLUSTRATING EFFECT OF OCCLUDED POINTCLOUDS ON POLICY LEARNING |
|-------------------------------------------------------------|----------------------------------------------------------|
| master chef can                                             | 1 2 3                                                    |
| vertices                                                   | 1 2 3                                                    |
| 1 cam                                                      | 1 2 3                                                    |
| 4 cam                                                      | 1 2 3                                                    |
| 4 cam+contact                                              | 1 2 3                                                    |
| pudding box                                               | 1 2 3                                                    |
| gelatin box                                                | 1 2 3                                                    |
| potted meat can                                            | 1 2 3                                                    |
| wood block                                                 | 1 2 3                                                    |
| foam brick                                                 | 1 2 3                                                    |
| bowl                                                       | 1 2 3                                                    |
| mug                                                        | 1 2 3                                                    |
| bleach cleaner                                             | 1 2 3                                                    |
| power drill                                                | 1 2 3                                                    |
| large marker                                               | 1 2 3                                                    |
| pitcher base                                               | 1 2 3                                                    |
| average                                                   | 1 2 3                                                    |

A. Dataset Generation Details

In real world experiments, we found Allegro hand is not strong enough to lift heavy objects like power drill or bleach cleaner, so we reduce all the object mass to less than 0.25 kg. During data randomization, we fixed Allegro hand torque force that is close to a real robot: 0.6 N.m, and randomize object mass $m \in [0.1 \text{ kg}, 0.25 \text{ kg}]$, and object friction coefficient $\mu \in [0.7, 0.85]$. We injected noise $\sim \mathcal{N}(0, 0.01) m$ to each candidate trajectory. In addition, we add 3 seconds of high-frequency perturbation to the palm at the end of the grasping trajectory to further identify unstable grasping poses. We use PyBullet [8] to generate the dataset and the robot is controlled at 12Hz in simulation.

B. Ablation on Input Representation

We aim to understand the impact that learning on noisy or occluded point clouds has on policy learning. In particular, we consider the impact of acquiring the point cloud using just a single camera versus using multiple cameras, as well as including the ability to sense binary contacts to deal with partial observability. In particular, we compare the following scenarios for pointcloud acquisition (as detailed in Table III):

**Full Vertices (Oracle):** This assumes no occlusion during robot movement, and the state is fully observable. Note that this will not be realistic for any real robot experiments, but it useful as a upper-bound on the performance of the partially observable case. On average, it achieves 65%, 75% and 79% success rate after three successive trials.

**One Camera:** If we only use one camera to render the scene, severe occlusion occurs when the robot is close to the
object, and two are on the side. We found that with 4 cameras, the policy is able to achieve similar performance as sampling from vertices: 61%, 70% and 74%.

**Binary Contact:** Occlusion is inevitable during real world experiments, we investigate whether adding extra information about per-finger binary contact could help the policy under partially observable conditions. For each finger tip link, we set a binary contact variable to 1—if the contact force is larger than 0.5N—and 0 otherwise. We concatenate this binary contact feature with robot poses as the input for the Kinematics Encoder. We found that this extra information can help to fill the gap in performance between 4 camera rendering and full vertices. In particular, it helps to improve performance on smaller objects, for example, tuna fish can, pudding box, gelatin box, and large maker, where occlusion usually occurs the most as shown in Table III. Interestingly, we found it performs slightly worse on some objects, for example master chef can and sugar box. We hypothesize the reason is that binary contact sometimes might mislead the network to think the grasp is stable enough to lift, and that a richer non-binary contact format could help with this issue.

C. Single Policy across Multiple Objects

We also show that we can train a single policy that can perform grasping across multiple objects in Figure 9. We group
objects with similar shapes: Cuboids and Cylinders. Each group contains 5 objects. Cuboid Objects include cracker box, sugar box, gelatin box, wood block and foam brick. Cylindrical objects include master chef can, tomato soup can, tuna fish can, mug and large marker. In addition, we select 5 objects that have diverse shapes. In order to evaluate the dropping performance when training across multiple diverse objects, we select 5 objects that yielded high performance when training per-object policies: master chef can, potted meat can, bowl, mug and large marker. We compute the average success rate from the per-object policies as an upper bound, and compare how the performance drops when training a single combined policy.

D. Qualitative Results

We show more qualitative results in both simulation (Fig.10) and real robot experiments (Fig.8). Our policy learns to adjust poses and fingers smoothly while approaching to the object. Further results can be found on the supplementary website https://sites.google.com/view/dextransfer/home