DiPGrasp: Parallel Local Searching for Efficient Differentiable Grasp Planning

Wenqiang Xu*1, Jieyi Zhang*1, Tutian Tang1, Zhenjun Yu1, Yutong Li1 and Cewu Lu1

Abstract—Grasp planning is an important task for robotic manipulation. Though it is a richly studied area, a standalone, fast, and differentiable grasp planner that can work with robot grippers of different DOFs has not been reported. In this work, we present DiPGrasp, a grasp planner that satisfies all these goals. DiPGrasp takes a force-closure geometric surface matching grasp quality metric. It adopts a gradient-based optimization scheme on the metric, which also considers parallel sampling and collision handling. This not only drastically accelerates the grasp search process over the object surface but also makes it differentiable. We apply DiPGrasp to three applications, namely grasp dataset construction, mask-conditioned planning, and pose refinement. For dataset generation, as a standalone planner, DiPGrasp has clear advantages over speed and quality compared with several classic planners. For maskconditioned planning, it can turn a 3D perception model into a 3D grasp detection model instantly. As a pose refiner, it can optimize the coarse grasp prediction from the neural network, as well as the neural network parameters. Finally, we conduct real-world experiments with the Barrett hand and Schunk SVH 5-finger hand. Video and supplementary materials can be viewed on our website: https://dipgrasp.robotflow. ai.

I. Introduction

Dexterous grasping is a long-standing problem in the robotics community. It is a task that achieves object grasp planning with high-DOF multi-finger robot grippers. Compared with the richly studied parallel-jaw grippers [1], [2], [3], [4], dexterous robot hands can perform more complex grasping [5], *e.g.*, human-like grasp. However, searching for a proper grasp pose in high-DOF configuration space is not as simple as the parallel-jaw grippers, since the latter only needs to consider the relative pose from the gripper wrist towards the object, while the former needs to determine the finger joint configurations.

Research on dexterous grasp planning lasts for decades [6], [7], [8], [9]. Methodologies followed by previous researchers can be roughly categorized into two main classes, *analytical* and *data-driven*. The analytical methods [6], [7], [8], [9] usually follow the model-based path and search for a grasp pose that can meet the requirements of some certain grasp quality metrics [6], [10]. Previous works on this track are generally slow, with a typical speed of $15s \sim 20$ min to generate one valid dexterous grasp pose. On the other hand,

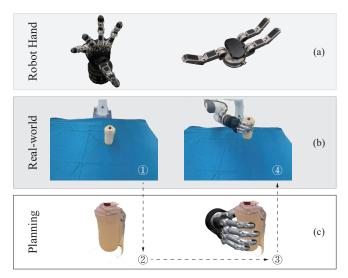


Fig. 1: DiPGrasp can (a) work with robot grippers with different DOFs. (b-c) It can produce high-DOF grasp poses efficiently from the observed point cloud and guide the execution in the real world.

data-driven methods leverage learning algorithms like deep learning [11], [12], [13] and reinforcement learning [14], [15] to predict grasp pose from noisy input of unseen objects. Once the network is properly trained, the inference time of grasp generation can be lowered to 30ms [11]. Methods on this track require a large amount of training data, which take considerable time (*e.g.*, 7 hours for 10K dexterous grasps in [16]) to generate.

Based on these observations, we determine a practical grasp planner should take the legacy of the conventional analytical path, but also can support the research on data-driven approaches. Thus, it should be *standalone*, *fast*, and *differentiable*. As a *standalone* planner, it can produce valid grasps based on a certain grasp metric and work with arbitrary grippers. As a *fast* planner, it can generate as many valid grasps as quickly as possible. As a *differentiable* planner, it takes gradient-based optimization techniques to solve the planning problem and can work with neural networks. To meet all these goals, we present **DiPGrasp**.

DiPGrasp is inspired by a geometry-based surface matching metric from prior research [8]. We add a force-based regularization term to enhance the grasp stability, and propose a novel force-based surface matching metric. To search the optimal grasps under this metric, DiPGrasp adopts the sample-optimize approach, initiating with sampled poses and refining them using the metric's objective function. Given

^{*}Equal contribution

¹{vinjohn, yi_eagle, tttang, jeffson-yu, davidliyutong, lucewu}@sjtu.edu.cn. Wenqiang Xu, Jieyi Zhang, Tutian Tang, Zhenjun Yu, Yutong Li are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. Cewu Lu is the corresponding author, a member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China.

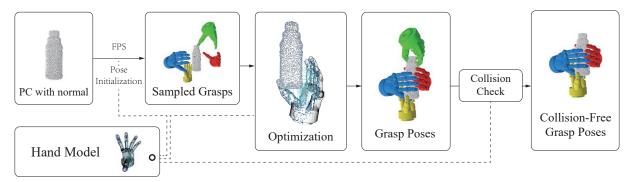


Fig. 2: DiPGrasp pipeline. DiPGrasp takes a point cloud with normal as input. It first samples locations on the point cloud (red dot) and initializes the pose accordingly. Then it operates the differentiable optimization process to generate the grasps.

the quadratic nature of the proposed force-based surface matching metric, it is amenable to gradient descent. By utilizing differentiable computing techniques [17], we efficiently batch sample poses and use gradient-based optimization like stochastic gradient descent (SGD). Both batch processing and gradient descent optimizers are commonplace in contemporary learning platforms [17]. This method greatly expedites the grasp search, allowing for simultaneous optimization at all sampled locations. In practice, DiPGrasp can perform a grasp search with a Schunk SVH hand using 80 initial poses in just 2.5s on an NVIDIA GeForce RTX 3080 GPU, using 8 GB memory, averaging \sim 118ms per valid grasp. This is substantially faster than EigenGrasp [7], which takes \sim 20s for a single grasp.

To prove the efficacy of the proposed DiPGrasp, we have applied it to three different applications: *Grasp dataset construction*, *Mask-conditioned planning*, and *Pose refinement*. Using DiPGrasp, we could construct a large dexterous hand dataset faster than the SOTA method [16] with a much higher valid proportion, generate valid grasp poses just on partial point cloud mask, or improve the quality of coarse poses generated by a neural network.

We summarize our contributions as follows:

- We introduce DiPGrasp, a differentiable, fast grasp planner compatible with robot grippers of varied DOFs. It employs gradient-based optimization for local grasp pose searches and can operate in parallel. This differentiability allows seamless integration into any differentiable frameworks.
- We use DiPGrasp for grasp dataset construction, mask-conditioned planning, and pose refinement. Real-world robot tests are conducted on mask-conditioned planning using models trained from the grasp dataset.

II. RELATED WORK

A. Analytical Dexterous Grasp Planner

Dexterous grasp planning research often employs heuristic metrics to assess grasp quality using a known object model, maximizing these metrics to find good grasp poses [6], [10]. Yet, real-world scenarios frequently involve objects with incomplete or unknown models. The high-DOF configuration space search for optimal grasp poses is non-convex, making

it challenging to locate the best solution. Consequently, optimization methods like simulated annealing [7], quadratic programming [8], and Bayesian optimization [9] are introduced, though they can be computationally time-consuming.

Regarding differentiable grasp planning, Liu et al. [18] introduce a differentiable ϵ -metric addressed with semidefinite programming, but it necessitates significant adjustments to serve as a standalone planner. The dataset for their study is sourced from GraspIt! [19]. Another approach by Liu et al. [9] employs a gradient-based method for grasp pose generation, but it is not very efficient as it yields limited successful grasps after lengthy computations. Grasp'D [20] offers a grasp synthesis process relying on differentiable physics simulation, but its results are sequential and demand a full object mesh. Contrarily, DiPGrasp can explore grasp poses concurrently and works with partial point clouds which could be generated from RGB-D observations instead of full meshes (See Fig. 1).

B. Data-driven Dexterous Grasp Planner

Data-driven grasp planning is increasingly recognized, with the capability to process incomplete object information, such as partial point clouds [11], RGB [15], RGB-D images [14], [13], and volumes [12]. These techniques, often underpinned by deep or reinforcement learning, can generalize to unknown objects with partial information.

For those employing deep neural networks, the common practice is either a generative [11], [12], [13] or discriminative [21], [13] task to predict grasps. However, given neural networks' prediction limitation in precision, several methods [22], [23] use a refinement step for improved accuracy. Some first propose indirect representations like heatmaps [24] or probabilistic distributions [13] from which grasp poses are derived

Incorporating reinforcement learning for dexterous grasping is challenging due to the complexity of dexterous configurations. Wu et al. [14] employ attention mechanisms for policy learning, while Mandikal and Grauman [15] utilize affordance data, eventually learning from videos with human actions [25].

III. FORCE-BASED SURFACE MATCHING METRIC

Previous works [8], [26] consider grasp planning as a surface-matching problem between the robot gripper and the object to be grasped. Such formulation can induce an intuitive optimization objective. However, it overlooks an important aspect of grasping, grasping stability. To amend that, we introduce a force-closure term in the optimization objective functions. In the following, we will first describe the geometry-based surface matching objectives, and then describe how to add the force-closure regularization into it.

A. Geometry-based Surface Matching

After a grasp is achieved, there will be multiple contacts between the gripper and object surfaces. Each contact i is defined by (S_i^f, S_i^o) , where S_i^f is the finger contact surface, and S_i^o is the object contact surface. S_i^f is a subset of *i*-th finger link surface $\partial \mathcal{F}_i$. $\partial \mathcal{F}_i$ is transformed by $\mathcal{T}(\cdot)$ given gripper pose $\mathcal{P}=(R,t,q)$, where $R\in SO(3), t\in\mathbb{R}^3$ are 6-DOF robot gripper wrist pose, and $q \in \mathbb{R}^k$ is the k-DOF finger pose. q_i is bounded by $q_{\min,i}$ and $q_{\max,i}$, which are the joint limits respectively. On the other hand, the object contact surface S_i^o is the nearest neighbor (NN) of S_i^f on the object surface $\partial \mathcal{O}$.

Then, we can formulate the grasp planning problem by searching the optimal grasp pose \mathcal{P} by minimizing the surface alignment error E:

$$\min_{R,t,q} \sum_{i=1}^{k} E(\mathcal{S}_i^f, \mathcal{S}_i^o) \tag{1}$$

s.t.
$$S_i^f \in \mathcal{T}(\partial \mathcal{F}_i; R, t, q)$$
 (2)

$$S_i^o = NN_{\partial \mathcal{O}}(S_i^f) \tag{3}$$

$$q_{min,i} \le q_i \le q_{max,i} \tag{4}$$

$$i = 1, \dots, k \tag{5}$$

The objective function can be reformed with two terms: point matching error E_p and normal alignment error E_n .

$$E_{sm}(R, t, q) = E_n(R, t, q) + E_n(R),$$
 (6)

$$E_p(R, t, q) = \sum_{i=1}^k \sum_{j=1}^m ||(x_{j_i} - y_{j_i})^T n_{j_i}^y||_2^2, \tag{7}$$

$$E_n(R) = \sum_{i=1}^m ||(Rn_{j_i}^x)^T n_{j_i}^y + 1||_2^2.$$
 (8)

 E_p measures the point-to-plane error between the j-th point on finger link i, x_{j_i} and the matched point y_{j_i} on object, $n_{j_i}^y$ is the normal vector at point y_{j_i} . $x_{j_i} \in S_i^f$, and thus is related to (R, t, q). E_n encourages the normals of the finger surface to align towards the normals of the object surface. A more detailed formulation description can be referred to [8].

B. Force-based Surface Matching

The surface-matching heuristics is intuitive and easy to optimize. However, it does not consider the force stability, thus the optimization objectives are inclined to generate an "in-contact" configuration rather than an "in-grasp" configuration.

To amend this, we introduce a variant of the force-closure term from [9], it can work with object models in point cloud form and a differentiable optimization scheme. We modify the point matching error in Eq. 7 to:

$$E_{fp}(R, t, q) = E_p(R, t, q) + ||Gc||_2,$$
 (9)

and

$$G = \begin{bmatrix} I_{3\times3} & I_{3\times3} & \dots & I_{3\times3} \\ \lfloor x_1 \rfloor_{\times} & \lfloor x_2 \rfloor_{\times} & \dots & \lfloor x_n \rfloor_{\times} \end{bmatrix}, \quad (10)$$

$$G = \begin{bmatrix} I_{3\times3} & I_{3\times3} & \dots & I_{3\times3} \\ \lfloor x_1 \rfloor_{\times} & \lfloor x_2 \rfloor_{\times} & \dots & \lfloor x_n \rfloor_{\times} \end{bmatrix}, \qquad (10)$$

$$\lfloor x_i \rfloor_{\times} = \begin{bmatrix} 0 & -x_i^{(3)} & x_i^{(2)} \\ x_i^{(3)} & 0 & -x_i^{(1)} \\ -x_i^{(2)} & x_i^{(1)} & 0 \end{bmatrix}. \qquad (11)$$

where x_i represents the contact point on hand and c_i denotes the corresponding normal vector of x_i . We utilize Farthest Point Sampling (FPS) to discern contact points, specifically by selecting four points (n = 4) from the palm side of the hand that fall within the top 20% of points nearest to the object. For the details of the force closure term please refer

Finally, we give the force-based surface matching heuristics:

$$E(R, t, q) = E_{fp}(R, t, q) + E_n(R).$$
 (12)

With this metric, we can design a grasp planner for arbitrary objects and robot hands.

IV. DIPGRASP

In this section, based on the force-based surface matching metric, we first describe how to optimize the objective function and make the process differentiable in Sec. IV-A. A collision-aware term to the grasp quality metrics E is introduced in Sec. IV-B. And the parallel sampling strategy in Sec. IV-C. Finally, we will describe an adjustable weighting map for different grasp type priors in Sec. IV-D.

The pseudocode of fully differentiable DiPGrasp is given in Algo. 1. The overall pipeline is illustrated in Fig. 2.

A. Optimizing Grasp Pose with A Gradient-based Solver

In this work, we directly use gradient descent for both wrist and finger pose optimization. Since R, t, q are the parameters to update, and as we can see from Algo. 1, all the operations (Line 6, 9-13) relevant to these parameters are differentiable. We can preserve the gradients for these parameters during iterations so that the gradient can be used to update the parameters of itself or a differentiable pipeline.

Gradient-based optimization is known to suffer from local minima [8]. But modern gradient descent-based optimizers (e.g., SGD [27]) usually adopt momentum or even secondorder momentum (e.g., Adam [28]) to try to escape from the

Algorithm 1: DiPGrasp

Input: Initial state R_0 , t_0 , q_0 , object surface $\partial \mathcal{O}$, gripper surface $\partial \mathcal{F}$, error threshold ϵ_0 , max iterations N_1 , N_2 Output: \hat{R}^* , \hat{t}^* , \hat{q}^* 1 $i_1 \leftarrow 0, i_2 \leftarrow 0;$ 2 $Loc \leftarrow sample(∂O); // Sec. IV-C$ $(R_s, t_s, q_s) \leftarrow initial_pose(Loc); // Sec. IV-C$ 4 Collision check; // Sec. IV-B 5 while $\Delta \epsilon \geq \epsilon_0$ and $i_1 \leq N_1$ do $S_{i_1}^f \leftarrow \mathcal{T}(\partial \mathcal{F} \otimes \mathcal{W}; R_s, t_s, q_s); // \text{ Sec. IV-D}$ $S_i^o \leftarrow NN_{\partial\mathcal{O}}(S_i^f);$ 7 while $\Delta \epsilon \geq \epsilon_0$ and $i_2 \leq N_2$ do // Sec. IV-A $\begin{array}{l} \mathcal{L} \leftarrow E^*(S_{i_1}^f, S_{i_1}^o); \\ dR, dt, dq \leftarrow \frac{\partial \mathcal{L}}{\partial R}, \frac{\partial \mathcal{L}}{\partial t}, \frac{\partial \mathcal{L}}{\partial q}; \end{array}$ 9 10 $R \leftarrow R - \alpha dR;$ 11 $t \leftarrow t - \beta dt$; 12 $q \leftarrow q - \gamma dq;$ $\Delta \epsilon \leftarrow |E^* - E_{prev}|;$ 13 14 end 15 16 end 17 Collision check (Sec. IV-B); // Sec. IV-B 18 $\hat{R}^*, \hat{t}^*, \hat{q}^* \leftarrow R, t, q;$

local minima. Thus, though previous gradient-based works [8] have proposed many complex optimization schemes, we find the commonly used SGD is stable enough.

B. Collision Handling

In the original surface matching heuristics, E does not consider avoiding collision during optimization. This will cause a large portion of the planned grasp pose to be in the collision and result in a failed grasp. Thus, in order to save more collision-free grasp poses, collision handling is desired.

In this section, we will introduce a differentiable barrier term E_b and a fast collision check method to avoid collisions.

a) Barrier Term: We first consider a distance measure between two points $d_i = (x_{j_i} - y_{j_i})^T (x_{j_i} - y_{j_i})$, and a barrier boundary \hat{d} . If two points are getting too close, the energy between them will grow exponentially. No repulsion will be applied if $d_i \geq \hat{d}$. Besides, we would like the barrier term to have at least C^1 continuity for gradient computation. With these ideas in mind, we borrow the definition of barrier term from [29]:

$$E_{b} = \frac{1}{m} \begin{cases} (d_{i} - \hat{d})^{2} \ln{(\frac{d_{i}}{\hat{d}})}, & 0 < d_{i} < \hat{d} \\ 0, & d_{i} \ge \hat{d} \end{cases}$$
(13)

This barrier function has C^2 continuity, which is sufficient for gradient descent. We set $\hat{d}=0.05$ for all experiments.

To note, the barrier term can also be used to prevent joints from updating to an out-of-range configuration. We

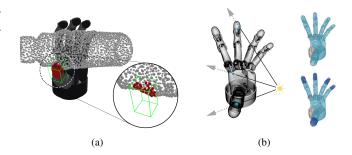


Fig. 3: (a) Collision check. Some points on the object surface are in collision (in red) with the bounding box, which represents the fingertip. (b) **Left**: Gripper weighting map can be automatically generated by ray casting. **Right**: Finger links like fingertips can easily be singled out according to the kinematic structure. (Darker area means bigger weight)

can define $d_{q_{min,i}} = |q_i - q_{min_i}|$ and $d_{q_{max,i}} = |q_i - q_{max_i}|$ to barrier the joint to get close to the limits. $\hat{d}_{min} = \hat{d}_{max} = (d_{max} - d_{min}) \times 0.15$. These terms give $E_{b,q_{min}}$ and $E_{b,q_{max}}$. We add E_b and $E_{b,q}$ to E and result in E^*

$$E^*(R, t, q) = E(R, t, q) + E_b(R, t, q) + E_{b,q}(R, t, q).$$
(14)

b) Collision Check: Though the barrier term can pull away the points from getting too close, it does not guarantee intersection-free results. Thus, after the optimization process, a collision check is essential for a better result. Since collision detection [29] is computationally expensive, here we adopt a simple collision detection approach.

As shown in Fig. 3(a), we represent each finger link and the palm as a bounding box. Then, we can tell if there are potential collisions by checking if any point is inside the bounding box. And we also apply this method to detect if self-collision happens. This step can be easily computed in parallel.

C. Parallel Sampling

In this section, we will describe the sampling strategy.

As shown in Algo. 1, the gradient-based solver reacts to one location at a time. Thus, we can utilize the "batchify" techniques which are commonly adopted in modern deep learning training [17] for parallel sampling and optimization. By sampling a bunch of initial poses and organizing them into a batch, we can search for valid grasp poses from these initial poses simultaneously. In this way, we can search K poses at the same runtime as 1 initial pose. The maximum of K is limited by the GPU memory and will be discussed in Sec. V.

For each sampled pose, we make the palm directly oriented to the sampled points, spread the finger, and place it at a distance of $d_{gripper}$ from the point. $d_{gripper}$ is different according to the grippers, as the finger lengths of different grippers may be different. For Barrett hand $d_{gripper}=15 \, \mathrm{cm}$, for Schunk SVH hand $d_{gripper}=12 \, \mathrm{cm}$.

We will first do a collision check to make sure the hand and object are not in collision with the initial pose. The extensive sampling can ensure multiple valid grasp outputs and the coverage of the object surface. We may have different strategies to select the one valid grasp to execute the plan. We will describe the grasp selection we use in Sec. V. Besides, we can also limit the sampling according to a certain constraint, e.g. mask on the surface. For example, an object mask from a scene, or an affordance mask from an object. We will give examples for the object mask in the application of **Mask-conditioned Planning**.

D. Gripper Weighting Map

It is obvious that we only want to align the object surface with the palmar side surface to form a grasp. If the dorsal side gets involved in the matching between the hand and object, the object might be aligned to the dorsal side, or trapped between the dorsal and palmar sides of the hand. However, there's no off-the-shelf algorithm that could separate the palmar side surface given a gripper (or a hand) model. One way is to define the palmar side surface of the gripper manually, which is laborious for new grippers.

Here, we describe a simple yet effective approach to annotating the palmar area of any grippers. As shown in Fig. 3(b) **Left**, inspired by the ray-casting pipeline [30], we place a light source in the palm, then cast large quantities of rays omnidirectionally. Then, we regard all the first-intersection points on the gripper surface for each ray as the palmar side points. Empirically, the location of the light source can be given by calculating the average location of all the fingertips. The whole process can be made automatically in this way.

Besides, we introduce a weight map to separate the different parts of the surface, which could make the grasp pose more diverse. For example, we could assign a bigger weight to the fingertip part while assigning a smaller to the other side, as shown in Fig. 3(b) **Right**. The fingertip is prone to getting closer to the surface, leading to a pinch grasp.

V. EXPERIMENTS

A. Task Definition

We set three tasks to show the efficiency and utility of DiPGrasp, namely **Grasp Dataset Construction**, **Mask-conditioned Planning**, and **Pose Refinement**.

a) Grasp Dataset Construction: We construct a comprehensive dataset for grasp analysis using RFUniverse [31] and plan to apply it across multiple applications. Our dataset includes 50 object models across five categories—bowl, box, sauce, tableware, and drink bottle—sourced from the AKB-48 dataset [32]. We select 8 models from each category for training and 2 for testing.

For robot grippers, we utilize the Barrett hand (7-DOF) and the Schunk SVH 5-finger hand (20-DOF). Our system can be compatible with various grippers, provided they have a URDF file.

To get the grasp poses for each object, we initially sample 2000 points on each object's surface using farthest point sampling (FPS), and apply DiPGrasp to determine potential grasp locations. These poses are later validated by RFUniverse [31]. A successful grasp was defined by the gripper's

ability to lift and hold the object 20cm above its start position for 3 seconds, even when interfered with randomly altered gravitational forces for 1 second.

Further, we construct scenes by randomly placing five grasp-annotated objects on a table. The grasp poses are transformed according to the object poses. We filter out any poses in collision with other objects. Each scene is captured using simulated RGB-D cameras, which employ IR-based depth rendering to minimize the sim-to-real depth discrepancy, aiding in training neural networks.

In total, our dataset comprises 2000 scenes, with 1800 designated for training and 200 for testing. For object grasp pose generation, the time efficiency of grasp searches varied between the Barrett and Schunk SVH hands, with the former requiring 25 minutes and the latter 50 minutes to process all 50 models. In terms of scene construction, each scene takes about 1 minute to generate. This process includes placing objects and filtering collided grasps, which could also be executed in parallel. Overall, the Schunk SVH hand yields 1.2 million collision-free grasps (1 million for training, 0.2 million for testing), while the Barrett hand produces 2.8 million (2.2 million for training, 0.6 million for testing).

b) Mask-conditioned Grasp Planning: Given a scene point cloud containing objects to grasp as input, we first use an off-the-shelf 3D point cloud instance segmentation algorithm to separate each instance point cloud, outputting instance masks. The training data are generated by taking RGB-D snapshots from three views in the scene, which then are merged into a scene point cloud. To mimic the real-world multi-camera calibration error, we augment the scene point cloud when merging multi-view RGB-D by giving them a random positional offset around 1cm.

To train Mask3D, we adopt the original hyper-parameters from its official implementation. Learning rate is 0.0001. Batch size is 4. The training epoch is 400. For each mask predicted by Mask3D, we use FPS to sample 50 points for pose initialization, and use DiPGrasp to generate grasp poses.

c) Pose Refinement: For each object instance mask, we could also use a neural network to generate grasp poses. We modify a PointNet++ [33] (called SimpleGrasp) to generate point-wise coarse grasp poses. We change the output layer of the PointNet++ to generate a (1+7+k)-d vector. 1-D for validness of the point (whether it is a good point to grasp). 7-d for the wrist pose which is represented by a 4-d quaternion vector and 3-d translation. k-d for the joint angle. Then, we could use the DiPGrasp to refine the coarse poses concurrently.

To train PointNet++, we use the exact hyper-parameters in its official implementation. Learning rate is 0.0001. Batch size is 8. The training epoch is 200. The optimizer is SGD with a momentum of 0.9.

The Grasp Dataset Construction task is designed to demonstrate the utility of DipGrasp as a standalone grasp planner. All three tasks reflect different procedures in a learning-based grasp detection pipeline.

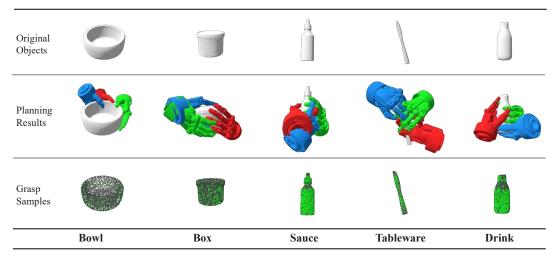


Fig. 4: Grasp planning results. **Upper**: Original object models. **Middle**: Sampled grasp planning results for visualization. The grasp poses in the display are randomly selected from the valid grasp poses in the **Bottom** row. **Bottom**: Distribution of the grasp planning results. Green means a valid grasp is found towards this point. Black means no valid grasp found.

B. Metrics

- a) ϵ -metric: ϵ -metric [6] is widely adopted to measure the force closure quality.
- b) Barrier-augmented Surface Matching (BSM) metric: Our planner is based on the barrier-augmented surface matching metric E^* . It can also be used to measure the grasp quality.
- c) Valid proportion: As our method can give multiple outputs simultaneously, we can examine the valid proportion for all the initial positions. The valid proportion is defined by the number of valid grasps over the number of initial positions. A valid grasp is a grasp pose generated when the optimization is converged, which does not necessarily guarantee successful execution in simulation or the real world. It is an important metric for dataset generation and coverage evaluation.
- d) Success rate: The success rate is defined by the number of successful grasps in simulation over the number of valid grasps generated. It takes into account the grasp selection and execution. We test the planned grasps in RFUniverse [31] and the real world to see the success rate.

C. Experiment results

1) Grasp Dataset Construction: For Barrett hand, over all categories, we can generate a grasp dataset with a 67.66% valid proportion after physics-based simulation. For Schunk SVH hand, we have a 26.5% valid proportion after physics-based simulation.

The reasons for the low valid proportion on the object surface are threefold: (1) It is natural that not every location is suitable for grasping. (2) The grasp search process is highly non-convex, making local minima inevitable. (3) Thin and small objects like tableware are hard to grasp stably in simulation.

However, since we have 2000 initial sample locations to search over the object surface, we still have over 1300 valid

grasps for Barrett hand, and over 500 valid grasps for Schunk hand. Besides, as searching at 80 locations will consume 8GB GPU memory, we have to split the search 25 times with our NVIDIA GeForce RTX 3080 GPU. Totally, it takes 40s for Barrett hand and 60s for Schunk hand to accomplish searching at 2000 locations. The average time for a valid grasp is 30ms and 118ms respectively. Even faster speed can be achieved with GPUs of larger memory, *e.g.*, RTX 3090 and A100.

We compare our method with previous methods regarding the quality and speed in Table I.

Method	ϵ	valid proportion (%)	ave. time (s)
ISF (B) [8]	0.159	61.9	2.13
EigenGrasp (B) [7]	0.237	/	18.75
DexGraspNet* [16]	0.1145	18	2.47
Ours (B)	0.201	67.66	0.030
Ours (S)	0.170	26.50	0.118

TABLE I: * is reported for reference, the dataset they generate has similar object shapes to ours, and the speed is measured with A100 GPU. We only compare ISF and EigenGrasp with Barrett (B) because the implementations of these methods do not support Schunk hand (S). EigenGrasp can only produce 1 grasp pose at a time, thus it does not have a valid proportion. We report the time it takes to produce the first valid grasp, and the time is measured on the same computational hardware setting.

2) Performance in Mask-conditioned Planning: We use Mask3D [34] as the segmentation module. For each instance mask, we get several valid grasp poses after the collision check. In simulation, we can test every valid grasp pose. But in the real world, it is time-consuming to verify every grasp, thus we select the grasp pose from the valid grasps with the minimum E^* , and check the success rate.

We only carry out the grasp planning for segmented objects with a confidence prediction larger than 0.7. The

scores in Table II are based on these detections. We compare our method with baseline methods ISF [8] and the grasp generation algorithm used in DexGraspNet [16] on real-world tests, and report the success rate in Table II. For the success rate of our method evaluated in the simulated scenes, please refer to the supplementary materials.

Env	tries	success	success rate
ISF (Barrett)	50	20	40%
DexGraspNet (Schunk)	50	11	22%
Ours (Barrett) without force	50	26	52%
Ours (Schunk) without force	50	22	44%
Ours (Barrett)	50	30	60%
Ours (Schunk)	50	24	48%

TABLE II: Success rate for mask-conditioned Planning with different planning algorithms and grippers in real worlds.

3) Performance in Pose Refinement: For pose refinement, we train SimpleGrasp with the segmented object point clouds and adopt the valid grasps generated by DiPGrasp as the corresponding grasp pose label. It is a common setting in grasp detection task [3].

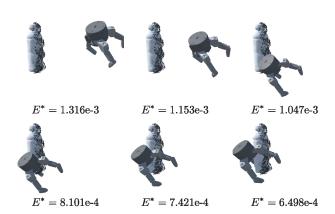


Fig. 5: Pose refinement. Even though the neural network prediction is extremely coarse, we still can progressively make it a better grasp with DiPGrasp.

a) Neural Network Refinement: Using gradient backpropagating from E^* , our differentiable planner can help optimize the parameters in SimpleGrasp neural networks. As our planner can be optimized with a gradient descent-based optimizer, they can be jointly trained with the same SGD optimizer. We train SimpleGrasp alone first, and then jointly train with DiPGrasp for the final 40 epochs. In comparison with training without DiPGrasp, incorporating DiPGrasp can improve the coarse grasp ϵ -metric from 0.085 to 0.101.

To note, DiPGrasp needs to join in the late stage of training, as there are few good initial poses for DiPGrasp to carry on in the early stage.

b) Pose Refinement: Apparently, though neural networks can predict good grasp poses sometimes, further refinement is needed in most cases. Our planner can make it with ease. In table III, we should assume no object model knowledge during the test stage. Therefore, we cannot calculate the ϵ -metric, and report the BSM-metric instead.

Method	BSM-metric ($\times 10^{-5}$)	speed (ms)
Barrett		
SimpleGrasp	863.61	132
3-step	148.44	332
6-step	90.08	532
10-step	67.43	812
13-step	60.32	1022
Schunk		
SimpleGrasp	3323.51	133
3-step	669.42	405
6-step	381.05	687
10-step	266.21	1059
13-step	227.39	1314

TABLE III: DiPGrasp can refine the pose predicted by SimpleGrasp.

Besides, we do not collect valid grasps, as the optimization process has not yet been completed. Thus, we report the total runtime (speed) instead of the average speed. A qualitative result is illustrated in Fig. 5 with Barrett hand.

D. Discussion

- 1) Failure Modes: Observing the failure examples in realworld experiments, we found that there are three typical failure modes:
 - The contact happens before the finger reaches the desired configuration, thus the object could be pushed far away. Such a mode often happens on the Barrett because the finger link is too long.
 - The contact surface is so smooth, it can lead to slip.
 Such mode often happens on the porcelain bowl with Schunk SVH hand.
 - The object is so thin that the gripper can not hold it. Such a mode often happens on the tableware.
- 2) Noise Sensitivity: Besides, as a geometry-based surface matching algorithm, it is imperative to evaluate how noise within the point cloud affects performance. To this end, we introduced Gaussian noise to the original point cloud at varying standard deviations and subsequently assessed the impact on algorithmic performance, with the results detailed in Table IV. It is evident that the valid proportion declines sharply when the standard deviation reaches 1 cm. Despite this, our algorithm remains sufficiently efficient on data with significant noise levels.

	no noise	$\sigma = 2$ mm	$\sigma = 5$ mm	$\sigma = 1$ cm
Barrett				
Valid Proportion(%)	67.66	64.32	55.98	37.45
Ave Time(s)	0.030	0.032	0.036	0.054
Schunk				
Valid Proportion(%)	26.50	25.11	21.21	14.70
Ave Time(s)	0.118	0.125	0.147	0.213

TABLE IV: Performance with noised data.

VI. CONCLUSION AND FUTURE WORKS

In this work, we introduce DiPGrasp, a standalone, fast, and differentiable grasp planner compatible with various robot gripper DOFs. DiPGrasp employs force-based surface-matching heuristics for less time-consuming gradient-based

optimization and uses parallel computing for simultaneous grasp searches. We've utilized DiPGrasp in grasp dataset construction, mask-conditioned planning, and pose refinement

As a differentiable planner, in the future, we aim to integrate it with other research areas such as learning to sample, adaptive weight mapping for grasp types, and optimizing objective functions. Additionally, we seek to create a differentiable dexterous manipulation framework using DiPGrasp.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Project of China (No. 2022ZD0160102) National Key Research and Development Project of China (No. 2021ZD0110704), Shanghai Artificial Intelligence Laboratory, XPLORER PRIZE grants.

REFERENCES

- [1] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1957–1964.
- [2] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.
- [3] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-Ibillion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.
- [4] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2901–2910.
- [5] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, "The grasp taxonomy of human grasp types," *IEEE Transactions on human-machine systems*, vol. 46, no. 1, pp. 66–77, 2015.
- [6] C. Ferrari and J. F. Canny, "Planning optimal grasps." in *ICRA*, vol. 3, no. 4, 1992, p. 6.
- [7] M. Ciocarlie, C. Goldfeder, and P. Allen, "Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem," in *Robotics: Science and systems manipulation workshop*sensing and adapting to the real world, 2007.
- [8] Y. Fan, H.-C. Lin, T. Tang, and M. Tomizuka, "Grasp planning for customized grippers by iterative surface fitting," in 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE). IEEE, 2018, pp. 28–34.
- [9] T. Liu, Z. Liu, Z. Jiao, Y. Zhu, and S.-C. Zhu, "Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 470–477, 2021.
- [10] B.-H. Kim, S.-R. Oh, B.-J. Yi, and I. H. Suh, "Optimal grasping based on non-dimensionalized performance indices," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium* (Cat. No. 01CH37180), vol. 2. IEEE, 2001, pp. 949–956.
- [11] V. Mayer, Q. Feng, J. Deng, Y. Shi, Z. Chen, and A. Knoll, "Ffhnet: Generating multi-fingered robotic grasps for unknown objects in real-time," in 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022, pp. 762–769.
- [12] D. Winkelbauer, B. Bäuml, M. Humt, N. Thuerey, and R. Triebel, "A two-stage learning architecture that generates high-quality grasps for a multi-fingered hand," in 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2022, pp. 4757–4764.
- [13] U. R. Aktas, C. Zhao, M. Kopicki, A. Leonardis, and J. L. Wyatt, "Deep dexterous grasping of novel objects from a single view," arXiv preprint arXiv:1908.04293, 2019.

- [14] B. Wu, I. Akinola, and P. K. Allen, "Pixel-attentive policy gradient for multi-fingered grasping in cluttered scenes," in 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2019, pp. 1789–1796.
- [15] P. Mandikal and K. Grauman, "Learning dexterous grasping with object-centric visual affordances," in 2021 IEEE international conference on robotics and automation (ICRA). IEEE, 2021, pp. 6169– 6176.
- [16] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang, "Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation," arXiv preprint arXiv:2210.02697, 2022.
- [17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," Advances in neural information processing systems, vol. 32, 2019.
- [18] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, "Deep differentiable grasp planner for high-dof grippers," arXiv preprint arXiv:2002.01530, 2020.
- [19] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.
- [20] D. Turpin, L. Wang, E. Heiden, Y.-C. Chen, M. Macklin, S. Tsogkas, S. Dickinson, and A. Garg, "Grasp'd: Differentiable contact-rich grasp synthesis for multi-fingered hands," in *Computer Vision–ECCV 2022:* 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI. Springer, 2022, pp. 201–221.
- [21] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, "Unigrasp: Learning a unified model to grasp with multifingered robotic hands," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2286–2293, 2020.
- [22] W. Wei, D. Li, P. Wang, Y. Li, W. Li, Y. Luo, and J. Zhong, "Dvgg: Deep variational grasp generation for dextrous manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1659–1666, 2022, publisher: IEEE.
- [23] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, "Generating grasp poses for a high-dof gripper using neural networks," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 1518–1525.
- [24] J. Varley, J. Weisz, J. Weiss, and P. Allen, "Generating multi-fingered robotic grasps via deep learning," in 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2015, pp. 4415–4420.
- [25] P. Mandikal and K. Grauman, "Dexvip: Learning dexterous grasping with human hand pose priors from video," in *Conference on Robot Learning*. PMLR, 2022, pp. 651–661.
- [26] Y. Fan, X. Zhu, and M. Tomizuka, "Optimization model for planning precision grasps with multi-fingered hands," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 1548–1554.
- [27] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv:1609.04747, 2016.
- [28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
- [29] M. Li, Z. Ferguson, T. Schneider, T. R. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: intersection-and inversion-free, large-deformation dynamics." ACM Trans. Graph., vol. 39, no. 4, p. 49, 2020.
- [30] H. Ray, H. Pfister, D. Silver, and T. A. Cook, "Ray casting architectures for volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 210–223, 1999.
- [31] H. Fu, W. Xu, R. Ye, H. Xue, Z. Yu, T. Tang, Y. Li, W. Du, J. Zhang, and C. Lu, "Demonstrating rfuniverse: A multiphysics simulation platform for embodied ai."
- [32] L. Liu, W. Xu, H. Fu, S. Qian, Y. Han, and C. Lu, "Akb-48: A real-world articulated object knowledge base," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," Advances in neural information processing systems, vol. 30, 2017.
- [34] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D for 3D Semantic Instance Segmentation," 2023.