Toward Human-Like Grasp: Dexterous Grasping via Semantic Representation of Object-Hand

Tianqiang Zhu Rina Wu Xiangbo Lin Yi Sun*
Dalian University of Technology, China
{zhutq,hswrn}@mail.dlut.edu.cn,{linxbo,lslwf}@dlut.edu.cn

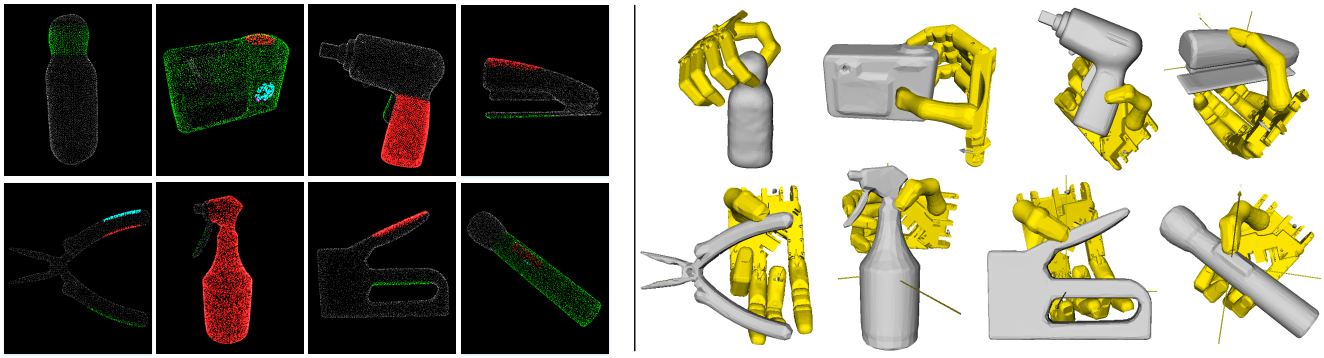


Figure 1: The left side shows the functional areas of the object, and each functional area is annotated with the semantic touch code to guide our model to generate the functional grasp of dexterous hand, which are shown in the right side.

Abstract

In recent years, many dexterous robotic hands have been designed to assist or replace human hands in executing various tasks. But how to teach them to perform dexterous operations like human hands is still a challenging task. In this paper, we propose a grasp synthesis framework to make robots grasp and manipulate objects like human beings. We first build a dataset by accurately segmenting the functional areas of the object and annotating semantic touch code for each functional area to guide the dexterous hand to complete the functional grasp and post-grasp manipulation. This dataset contains 18 categories of 129 objects selected from four datasets, and 15 people participated in data annotation. Then we carefully design four loss functions to constrain the model, which successfully generates the functional grasp of dexterous hand under the guidance of semantic touch code. The thorough experiments in synthetic data show our model can robustly generate functional grasp, even for objects that the model has not see before.

1. Introduction

We have been pursuing to make robots grasp and manipulate objects like human beings, so as to help us complete various tasks. Although there are already some dexterous manipulators like human hands, it is very challenging to control them to operate like a human. Generally, the methods of dexterous grasp synthesis are mainly divided into hand-centered methods and object-centered methods. Previous work in hand-centered methods has focused on recording grasping activity in the form of hand joint configuration or grasp types by landmark tracking [19], model based or data-driven based hand pose estimation [6, 18]. However, due to the high degree of freedom, self-occlusion and self-similarity among fingers, it's challenging to accurately annotate the coordinates or angles of each hand joint.

Recently, researchers begin to shift their focus to the object-centered grasp synthesis. Their methods pay more attention to the pose [10, 34], shape [28, 39, 42, 43, 46–48, 50, 51, 59] and function [5, 16, 17, 23, 36, 44, 45, 60] of the object for grasp and manipulation, observing where contact between the object and the human hand. GraspIt! [37] is used to predict feasible grasps for the object geometry with force closure condition, but it cannot guarantee to generate anthropomorphic grasps. In order to predict human-like grasps, it should be aware of where objects can be grasped by a dexterous hand. ContactDB [4], as the first large-scale dataset that records detailed contact maps for human grasps of household objects, synthesizes the grasp from the contact map and does not need manual specification of per-finger

contact point. However, since each part of the hand does not correspond to the specific part of the contact map, multiple hand configurations may sometimes result in the same contact pattern in this method. UniGrasp [52] considers both the object geometry and gripper attributes as inputs to select a set of contact points on the surface of the object such that these contact points satisfy the force closure condition and are reachable by the gripper without collisions, but it's hard to precisely make contact with these points under noise.

In contrast, our approach uses the real functional area of the object to guide the generation of functional grasp, avoiding the "multiple to one" issue caused by the contact map and the difficulty of predicting contact points. Specifically, we segment the functional parts of the object for grasping, and record touch code to describe whether fingers or palm of the hand contact each part of the object. In this case, the positions of the hand are optimized to touch the object 3D surface. Furthermore, the proposed high-level semantic touch code does not require low-level annotation of joint coordinates or angles of the hand which greatly simplifies the hand pose annotation in the grasping tasks. And most importantly, the touch code also includes a grasp functional intent that guides the fingers towards the task oriented human-like grasp. Given both the segmented functional parts on the object surface and the semantic touch code as input, we train a deep learning model to seek a human-like grasp that gracefully fit the functional parts and the specification of the touch code. The proposed dataset and grasp synthesis results are illustrated in Fig. 1. Moreover, we also train a semantic segmentation network to predict the functional areas and semantic code of objects. At the same time, we connect the segmentation network with the aforementioned grasp synthesis network, and realized the functional grasp synthesis based on the object point cloud on the test set. Code and data will be provided here. The contributions of this paper can be summarized as follows.

- We propose a grasp synthesis framework that considers both the object functional parts and touch code of the dexterous manipulators for each part of the object. This joint representation of object-hand interaction is compact and effective which leads to successful task-oriented grasps like human hands.
- We create a dataset to train our network. This dataset contains 129 objects in 18 categories selected from 4 data sets. For each object, we segment the functional areas of the object, and annotate the touch code for each functional area to guide the dexterous hand to complete the functional grasp and post-grasp task.
- We have validated our method in synthetic data and showed that our model can also generate human-like grasps. This proves the importance of the proposed semantic object-hand representation on grasping.

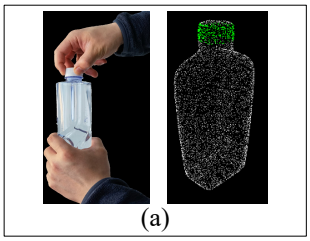
2. Related Work

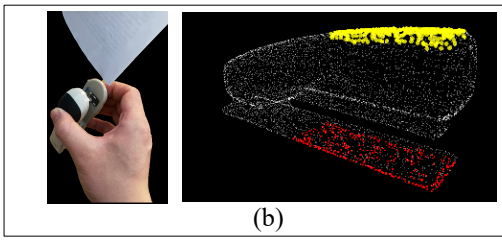
Dexterous Robotic Grasping In recent years, many researches have been devoted to the grasp planning of dexterous manipulators with high degrees of freedom [3, 10, 11, 14, 27, 31–33, 52]. Generally, these researches are mainly divided into hand-centered methods and object-centered methods. Most hand-centered methods [16, 23, 44, 45, 60] focus on conveying the grasp intention to the robot through human demonstrations. Among them, video [20] or kinesthetic demos [21, 53, 54] are used to record human demonstrations, data gloves [2, 13, 15] or visual recognition [49] are used to capture hand pose. However, to map human manipulation to different end-effector, different mapping methods need to be designed, and mapping rules should be carefully adjusted to be suitable for a variety of scenarios.

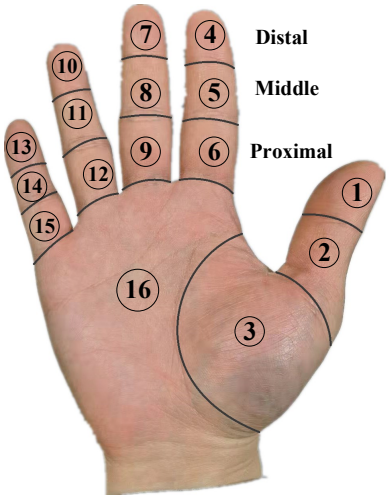
Recently, researchers begin to shift their focus to the object-centered grasp synthesis [14, 27, 32, 33]. [12, 22, 38, 57] guide the robot to complete the manipulation task by specifying the contactable area of the object, but this is not enough to teach the robot to achieve the fine-grained and dexterous manipulation. ContactGrasp [5] uses the contact heatmap in ContactDB [4] as grasp affordances, and online optimizes the stable grasps generated by GraspIt! [37] to match the contact heatmap, so as to obtain a human-like grasp. Similarly, [36] uses a deep reinforcement learning (RL) model in the simulation environment to generate grasp that conform to the contact heatmap. However, due to the size of contact heatmap depends on the human hand, it is unavoidable that the contact heatmap may be larger than the functional area of the object, and there is no clear contact guidance for the contact heatmap. Therefore, it may lead to finger contact outside the functional area of the object, and even make the wrong part of the hand touch the functional area of the object. In contrast, we accurately segment the functional areas of objects, and use concise and clear semantic code to guide the dexterous hand to interact with objects like human.

Grasp affordances Grasp affordances is generally centered on the object, and is used to guide the interaction between the manipulator and the object. Previous researches mainly focused on marking the contactable areas of objects in images [9, 12, 23, 26, 29, 30, 35, 41], but it is too rough to guide a dexterous hand to complete fine-grained manipulation tasks. Recently, with the development of deep learning in the field of computer vision, [10, 32, 33] proposed 3D pose-based affordances by marking configurations of specific manipulator during grasping. However, these methods are specific to the labeled manipulator, so that difficult to be applied to other dexterous hands. ContactDB [4] presents the contact heatmap of hand-object interaction on the object as grasp affordances, and ContactPose [6] further annotates the hand pose during the interaction. But as mentioned above, because the contact heatmap is derived from

Figure	Points Color	Touch code																Intention code			
		Thumb			Index			Middle			Ring			Little			Palm	Press	Click	Slide	Firmly touch
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	A	B	C	D
(a)	White	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Green	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
(b)	White	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Yellow	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	Red	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0


(a)


(b)



(c)

Figure 2: **(a)** describes the demonstration of the participant and the segmentation of object functional areas when opening the bottle. **(b)** is another example. The 16 parts of hand in **(c)** correspond to the 16 bits of the 'touch code' in turn. The **table** records the 20-bit semantic code of each functional area of the object in (a) and (b)

the human hand, it may not match the functional areas of the object, and may not applicable to other manipulators. In contrast, we perform fine-grained semantic segmentation of objects according to their functions and structures, and use the annotated touch code to guide the manipulator to correctly contact the functional areas of the object.

3. Method

To make dexterous hands grasp and manipulate objects like human beings, it is necessary to let the robot understand which parts of the object should be grasped under the current task, which hand parts should be used to contact these object parts, and what kind of action should be taken after the contact. Therefore, in order to achieve the above three requirements, this paper proposes a new dataset, which segments the functional parts of the object for grasping to intuitively show the manipulator which object parts can be grasped, and uses explicit semantic annotation to tell the dexterous hand which finger or palm should contact these parts, and what kind of actions should be taken after completing the predetermined grasp. Moreover, we propose a deep learning approach to generate functional grasp of dexterous hand, which uses the above-mentioned semantic information as a object-hand interaction constraint. Because the main purpose of this paper is to propose a new dataset and discuss how to use the dataset to guide dexterous hand to generate functional grasp, we present the semantic segmentation network in the supplementary material due to limited space. This section is organized as follows: **Section 3.1** describes our dataset in detail, **Section 3.2** describes how to achieve functional grasp, and **Section 3.3** describes the loss function used in our approach.

3.1. Dataset of Object-Hand Semantic Representation for Grasping

Many household objects are designed to be used by human hand, and to correctly use an object often requires us to touch specific parts of this object, which we call the functional area of the object. Meanwhile, the correct grasping or manipulation is also inseparable from the cooperation of the hand, which needs to use specific parts of the hand to touch these functional areas [8], and then performs specific actions. For example, as shown in Fig. 2(a), when opening a bottle, most people will grasp the bottle with their left hand, while the thumb, index finger and middle finger of the right hand will firmly grasp the bottle cap. Based on this observation, we propose our semantic segmentation dataset, which contains the interaction between the right hand and the object when the object is manipulated. As for other situations, such as the interaction between the left hand and the object will be our future research work. The generating process of the dataset is as follows.

We select 129 objects in 18 categories with clear usage from four commonly used object datasets, namely BigBIRD [55], KIT Object Models Database [25], YCB Benchmarks [7], and Grasp Database [24], where meshes of various household objects are included. The 18 categories of objects are electric drill, clamp, hammer, flashlights, bottles, screwdriver, scissor, spatula, spray can, spray bottle, banana, pitcher, camera, dumbbell, light bulb, pliers, nail

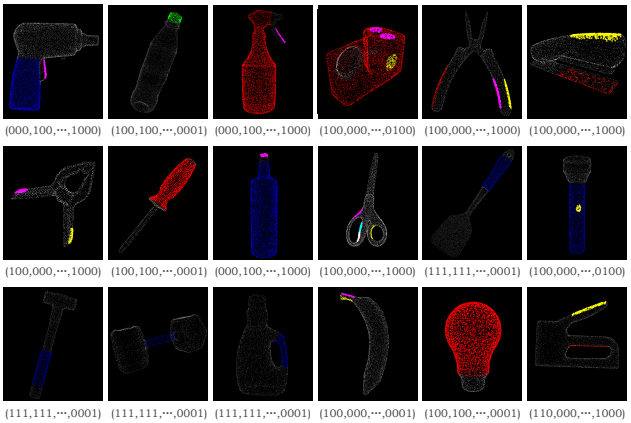


Figure 3: A brief presentation of the segmentation of functional areas and 'touch code' of 18 categories of objects in our dataset.

gun, and stapler. We perform segmentation according to functional parts of the objects. The functional parts are determined either according to user's manual of this object, or by votes of 15 participants who are able-bodied males and females between the ages of 20 and 65. To ensure the accuracy and consistency of semantic segmentation of object functional parts, two PhD students use the 'semantic-segmentation-editor' software to annotate and check together. Finally, one segmentation scheme remains for one task per object.

Specifically, to clearly express the grasping behavior between hand and object, as shown in Fig. 2, each functional part is labeled with 16-bit 'touch code' indicating whether it can be touched by corresponding hand part (Fig. 2(c)) and 4-bit 'intention code' indicating four action intentions of press, click, slide and firmly touch after contact. For example, the stapler shown in Fig. 2(b) is segmented into the head (yellow), the base (red) and the other parts (white). And the code of the stapler head is '$100,000,000,000,000,0|1000$', indicating that the robot should touch the stapler head by its distal thumb, and then perform a pressing operation. The 20-bit semantic code is necessary and important for the dexterous hand to learn to manipulation like human hands, because the first 16-bit 'touch code' can guide the dexterous hand to correctly touch the functional area of the object, so as to guide the synthesis of functional grasp, and the last 4-bit 'intention code' can tell each hand part what post-grasp action needs to be performed, so that the object can be used correctly. Therefore, although this paper only discusses how to use 'touch code' to generate functional grasp, in order to facilitate subsequent research and ensure the integrity of the dataset, we annotate both 'touch code' and 'intention code' in the dataset.

In summary, our dataset describes the necessary conditions for manipulating objects with intuitive segmentation and concise and clear semantic touch code. As shown in Fig. 3, there are 18 categories of objects, and the functional areas of each object are segmented and annotated with 20-bit semantic code.

3.2. Network structure

Functional grasp synthesis is a challenging research task in robotics [5]. A good functional grasp means that the dexterous hand needs to correctly touch the functional area of the object, so as to facilitate to perform the post-grasp action, such as using electric drill, the index finger should touch the On/Off switch while other parts of hand holding the drill handle. In this section, we will describe how to use the 16-bit 'touch code' we annotated to guide the network to generate the functional grasp of ShadowHand, without laborious annotation of hand pose [6], nor to use the grasp synthesis and feedback of the simulation software [36]. We formulate the process of functional grasp synthesis as follows: given an object point cloud O, and the 'touch code' C, we train a model $\mathcal{M}$ which outputs the configurations of ShadowHand to achieve functional grasp. The hand configurations include the angle of each joint (J), as well as the spatial rotation (R) and translation (T) of the hand relative to the object. The overall formulation is:

$$\mathcal{M} : \{O, C\} \Rightarrow \{J, R, T\} \tag{1}$$

As shown in Fig. 4, we concatenate the original point cloud of the object with the 16-bit 'touch code' as the input of the model. The input data first passes through the point cloud feature extraction network built with kernel point convolution (KPConv) [56], which realizes the convolution operation on the unordered point cloud by designing a spherical convolution kernel. Subsequently, the features extracted by the point cloud feature extraction network are fed into the fully connected layers, which outputs the hand configurations$\{J,R,T\}$. To avoid our model from trapping in local optimum, while improving the stability of the generated grasp, we pre-train the model with the grasp data from [32, 33], which generates adequate and stable grasps of ShadowHand with GraspIt! [37]. However, it is obvious that pre-training can only ensure that the grasp generated by the network is stable, which is far from the requirements of functional grasp with the functional areas of the object to be touched by specific parts of the dexterous hand.

To meet the basic requirement of functional grasp, we propose the attraction loss ($L_{attraction}$), which matches each functional area with the parts of the dexterous hand that need to touch this functional area according to the 16-bit 'touch code', and then minimizes the distance between them. To make the 16-bit 'touch code' applicable to ShadowHand, we select 16 from the 21 links of ShadowHand as shown in Fig. 5(a), which correspond to the 16 parts of the human hand in Fig. 2(c). Then, in order to calculate the

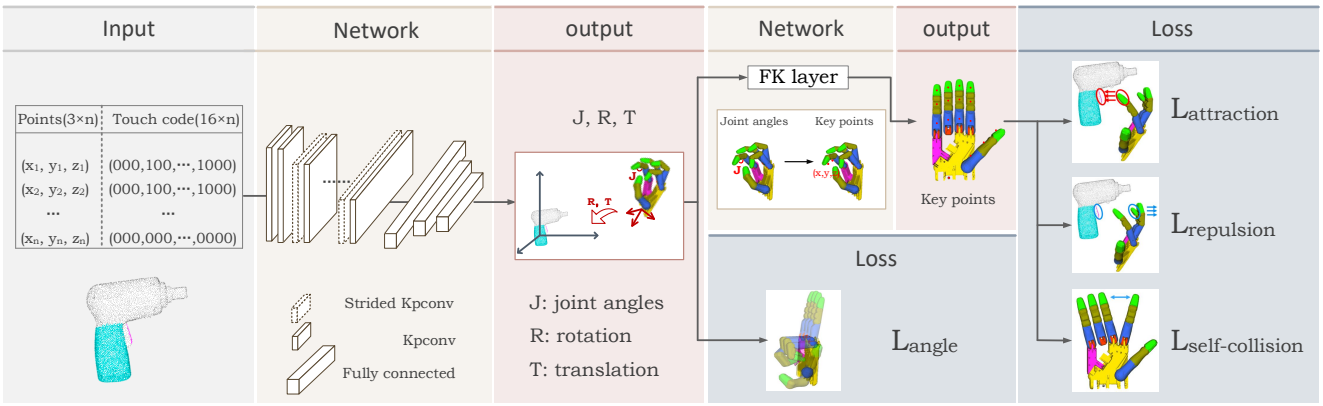


Figure 4: The overall architecture of our functional grasp synthesis framework. The original point cloud of the object with the 16-bit 'touch code' is fed into the network, which generates the configurations of the hand that conform to the functional grasp under the guidance of four loss functions.

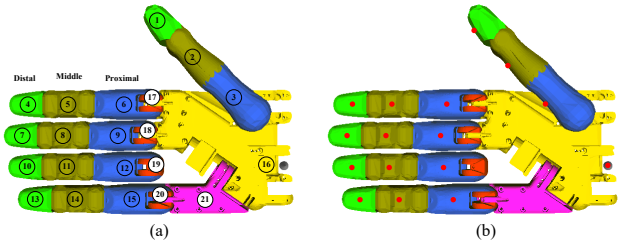


Figure 5: The 21 links of ShadowHand are marked as shown in (a), and 1 to 16 links are selected that correspond to 16 parts of the human hand in Fig. 2. (b) shows the 16 key points for calculating the loss function on the selected links.

distance, we select 16 keypoints of 16 hand links on the inner surface of the palm as shown in Fig. 5(b), and represent the distance between the link and the functional area by the minimum distance between the link's keypoint and the point set of the object functional area. The necessary condition of using these keypoints to calculate the distance is that their spatial coordinates need to be obtained in real time during the model training process. To this end, we use Pytorch [40] to write a differentiable forward kinematics derivation (*FK*) layer of ShadowHand, which can calculate the spatial coordinates of each key point according to the dexterous hand configurations$\{J,R,T\}$. A similar work of Tensorflow [1] version can be seen in [58]. In this way, we can force the hand links to close to corresponding functional area by minimizing $L_{attraction}$, which is the weighted sum of those distances between each link's keypoint and its corresponding functional area. However, due to the high degree of freedom of ShadowHand, the following problems may arise when only $L_{attraction}$ is used to constrain model to generate functional grasp: i) Those links that are not constrained by $L_{attraction}$ may appear anywhere. Such as opening a bottle, $L_{attraction}$ can only ensure that the distal thumb, distal index finger and distal middle finger are on the bottle cap, but the remaining links may appear inside the bottle; ii) $L_{attraction}$ cannot limit the generated joint angle of dexterous hand, so they may exceed their motion limits, resulting in an unreasonable hand shape; iii) Some functional areas can be touched by multiple links at the same time, such as bottle cap, so it may happen that multiple links approach the same point, which leads to self-collision of the robot hand.

To solve the above three problems, we propose the following solutions respectively. For the first problem, we propose repulsion loss ($L_{repulsion}$), which aims to keep each link away from the part of the object that does not correspond to this link. As shown in Fig. 4, while the switch of the electric drill attracts the distal index finger through $L_{attraction}$, it also repels the remaining hand links through $L_{repulsion}$. In this way, those links that are not constrained by $L_{attraction}$ will be constrained by $L_{repulsion}$. Meanwhile, since all the links are on the same kinematic chain, the hand pose can be basically constrained by the combination of attraction loss and repulsion loss.

For the second problem, because each joint of the robotic hand has a limited range of motion, we propose angle loss L_{angle}) to keep the generated joint angle θ_n between its lower bound θ_n^{min} and upper bound θ_n^{max}. As for the third problem, it can be solved by keeping the links away from each other, so that the self-collision avoidance loss ($L_{self-collision}$) is proposed. The details of these four loss functions are described below.

3.3. Loss Functions

As mentioned before, ***attraction loss*** ($L_{attraction}$) is the weighted sum of the distances between the point set of each object functional area and its corresponding link's keypoints. In the training process, we force the hand links to approach their corresponding functional areas by making $L_{attraction}$ close to 0. The $L_{attraction}$ is defined as follow:

$$L_{attraction} = \sum_{i=1}^{N} \sum_{j=1}^{16} \alpha_j \cdot dis\left(k_j, o_{ij}\right)) \tag{2}$$

where, in each training batch, N objects' data are fed into the model. α_j is the weight determined by the importance of each link. For example, the thumb is used the most in daily manipulation, so we set a higher weight for it. And $dis\left(k_j, o_{ij}\right)$ represents the minimum Euclidean distance between each link' kepoints and the point set of its object functional area, in which k_j is the key point on the $j\text{-}th$ hand link, and o_{ij} is the point set of the functional area corresponding to the $j\text{-}th$ hand link on the $i\text{-}th$ object.

Repulsion loss ($L_{repulsion}$) restricts the links that are not constrained by $L_{attraction}$ through exclusion. As shown in the following formula:

$$L_{repulsion} = \sum_{i=1}^{N} \sum_{j=1}^{16} \gamma_j \cdot max\left(log \frac{\beta_j}{dis\left(k_j, \widetilde{o_{ij}}\right) + \varepsilon}, 0\right) \tag{3}$$

where γ_j is the weight for each link. The max function and the threshold β_j can realize that when the $dis\left(k_j, \widetilde{o_{ij}}\right)$ exceeds the threshold β_j, the punishment for the $j\text{-}th$ link will be reduced to 0. And the $dis\left(k_j, \widetilde{o_{ij}}\right)$ represents the Euclidean distance between the $j\text{-}th$ link's key point k_j and the part $\widetilde{o_{ij}}$ of the $i\text{-}th$ object that the link needs to stay away from. Here, we adopt a logarithmic (log) function to punish this distance based on its properties that the punishment gets stronger as the distance diminishes, while the punishment drops rapidly as the distance slightly increases. In this way, it can not only play a good repulsive effect, but also help to balance with $L_{attraction}$, so as not to repel the hand far away from the object. Besides, ε is a small positive number to prevent the denominator from being 0.

Angle loss (L_{angle}) is a linear punishment for the angle θ_n generated by the model, keeping it between its lower bound θ_n^{min} and upper bound θ_n^{max}. The expression of angle loss is as follow, where N is the total number of joints:

$$L_{angle} = \sum_{n=1}^{N} max\left(\theta_n - \theta_n^{max}, 0\right) + max\left(\theta_n^{min} - \theta_n, 0\right) \tag{4}$$

The function of the ***self-collision avoidance loss*** ($L_{self-collision}$) is to keep the links away from each other, and the expression of which is as follow:

$$L_{self\_collision} = \sum_{i=1}^{16} \sum_{j=1}^{16} \mu_{ij} \cdot max\left(\delta_{ij} - dis\left(k_i, k_j\right), 0\right) \tag{5}$$

where δ_{ij} is the distance threshold to ensure that when the Euclidean distance between the ith and jth links exceeds δ_{ij}, punishment won't occur. And μ_{ij} is the weight.

Finally, *attraction loss*, *repulsion loss*, *angle loss* and *self-collision avoidance loss* are linearly combined into the final loss L, which is minimized in the training process:

$$\begin{aligned} L = & \lambda_1 \cdot L_{attraction} + \lambda_2 \cdot L_{repulsion} \\ & + \lambda_3 \cdot L_{angle} + \lambda_4 \cdot L_{self\_collision} \end{aligned} \tag{6}$$

4. Results

As mentioned in Section 3.2, we take the semantic information of the object-hand interaction as a guide, and lead the model to generate functional grasp for the dexterous hand through well-designed pre-training and loss functions. This section will evaluate our results through experiments. After a brief introduction of the experimental setup, we will present how our proposed loss functions lead the model to generate functional grasp step by step, and the impact of pre-training on the final generated grasp in section 4.1. In Section 4.2, we will further discuss the performance and existing problems of the trained grasp synthesis model in the training set, as well as show the grasp results generated on the test set after connecting the semantic segmentation network and grasp synthesis network.

We split the set of 129 labeled objects into a 104 training set and a 25 testing set. All experiments are carried out on a desktop with an Intel® Core^{TM} i7-7700 CPU @ 3.60GHz × 8, 32GB RAM, and an NVIDIA® Geforce GTX 1080 graphics card with 8GB memory, on which training the model presented in this paper takes above 24 hours.

4.1. Ablation Studies

In section 3.2, we describe in detail the role of pre-training and the four loss functions including attraction loss, repulsion loss, angle loss and self-collision avoidance loss. Next, we will verify the contribution of each component through experiments.

Fig. 6 shows some functional grasps generated by constraining the network with different combinations of 4 loss functions after pre-training. The first row of images shows the segmented parts of each object for grasp, the second row of images presents the results of using only $L_{attraction}$. It can be seen that the hand links are trying to get close to the functional areas of the object, such as grasping stapler in the first image of Fig. 6 with distal thumb pointing to the head of the stapler while the distal of the remaining four fingers are close around the bottom of the stapler. However, as analysed in section 3.2, those links that are not constrained by $L_{attraction}$ will run into the object, such as most part of the hand entering the interior of the stapler. With the addition of $L_{repulsion}$, as shown in the third row of Fig. 6, the hand basically run out of the object. But it can be seen that the hand shape is very strange. For example, the grasp of pliers on the fourth image of this row, the little finger has unreasonable joint angles, and several links collide. With the addition of L_{angle} and $L_{self-collision}$ in sequence, as shown in the fourth and fifth rows, it can be seen that the

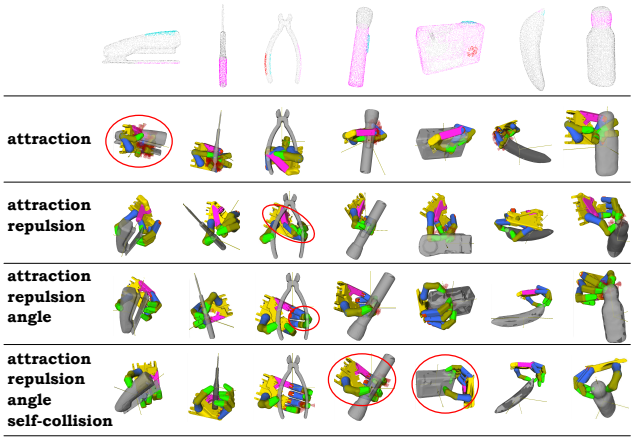


Figure 6: The first row shows the function areas of each object, while the second to the fifth rows show the generated grasp results of the network after adding *attraction loss*, *repulsion loss*, *angle loss* and *self-collision avoidance loss* these four loss functions in turn.

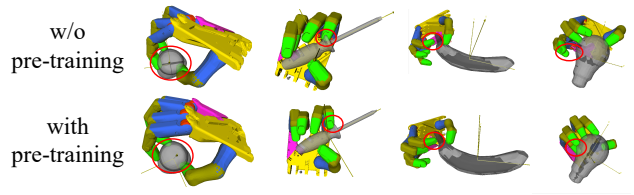


Figure 7: The comparison of the grasp results with or without pre-training.

joint angles of the dexterous hand in the fourth row are within a reasonable range, and the hands in the fifth row do not have self-collision. Meanwhile, the hand poses in the last row satisfy the expected functional grasps. For example, to use the flashlight in the fifth column, the distal thumb presses the switch, while the remaining links grasp the flashlight body, or to use the camera in seventh column, the distal thumb presses the function button, and the distal index finger presses the shutter button at the same time.

For **pre-training**, which is used to initialize the network with good parameter and enable the network to generate more stable grasps, we use two numerical metrics to illustrate its function. The first is the average distance between the object function areas and their corresponding links' key points. The smaller the distance is, the closer the hand link to its functional area. We abbreviate this distance as Mean DoC. The second is QM (Grasp Quality Metrics) calculated by GraspIt! [37], which refers to the ability of a grasp to resist interference from external forces. When the grasp is stable, the QM value is between 0 and 1, and the higher the value is, the more robust the grasp is. When the grasp is unstable, the value is -1. Specifically, we use QM>0 to represent the rate of stable grasp, and the average of all stable grasp's QM (Mean QM) to represent the overall stability.

Pre-train	Dataset	Mean Doc(cm)	QM>0	Mean QM
without	Train	1.24	65.05%	0.032
	Test	1.73	60.0%	0.029
with	Train	0.62	81.73%	0.081
	Test	1.06	80.0%	0.059

Table 1: Mean DoC is the average distance between the object function areas and their corresponding links' key points, while the smaller the Mean DoC is, the closer the hand links to their functional areas; QM is used to measure the robustness of a grasp. When QM>0, the grasp is stable, and the higher the QM value, the more robust the grasp is. Here, we use QM>0 to represent the rate of stable grasp and Mean QM to measure the stability of all grasps.

As shown in Tab. 1, it can be seen from Mean DoC that the average distance between the functional area and their corresponding links becomes smaller after pre-training. This is because after pre-training, as shown in Fig. 7, the dexterous hand is more inclined to contact the surface of the object, which alleviates the situation of penetrating or staying away from the object. From QM>0 and mean QM, we can see that the model can generate more and better stable grasps after pre-training. The above observations are consistent with our original intention of using pre-training.

4.2. Experimental Results and Discussion

In the section 4.2, we prove that the semantic segmentation of our annotation can effectively guide the model to generate the functional grasp of dexterous hand, and also prove that the proposed constraint functions and training methods have their own important roles. In this section, we will further analyze the results generated by our model.

As mentioned earlier, our model is trained without accurate hand annotation of functional grasp. Instead, it is trained by our proposed dataset and four well-designed loss to make the model try to fit the correct functional grasp. Therefore, this section first analyzes the performance of the network on the training set, and then discusses whether the grasp generation network can be generalized to objects that have not been seen by the grasp network and whose functional areas and semantic code are predicted by the segmentation network.

Fig. 8 shows some representative results on the training set. It is obviously that for most objects, the functional grasps we generate are in line with humans' habits of manipulating these objects. For example, the first image in Fig. 8 shows the use of nail gun. The grasp in this figure is very helpful to the execution of the post-grasp action to complete the binding task. Compared with the research work aiming at the force closure of dexterous hand, as shown in Fig. 9, it can be found that our method explic-

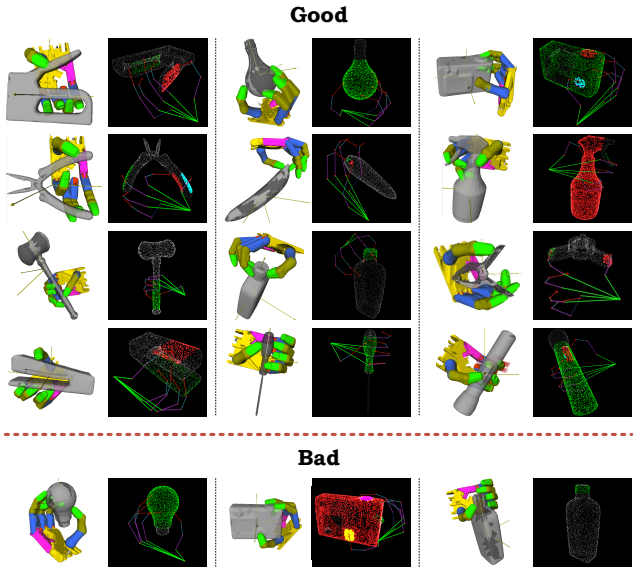


Figure 8: The quality results of functional grasp about the object in training set.

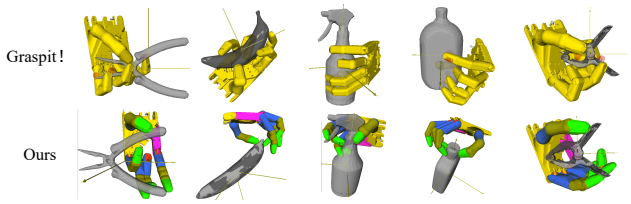


Figure 9: Comparisons between GraspIt! and ours.

itly directs the dexterous hand to touch the functional areas of the object. For example, when using the pliers, as shown in the first image in the second row of Fig. 8, our method can ensure that the two handles of the pliers can be correctly touched by different fingers. In contrast, the hand in Fig. 9 grasps the head of the pliers.

However, there are still some problems in our method, such as grasping the bulb in the first image in the last row of Fig. 8. Although all the fingertips are on the bulb, the bulb thread is facing the palm of the hand, so the operation of installing the bulb cannot be completed. To solve this problem, we plan to further constrain the approaching direction of dexterous hand in future work.

In order to further verify the practicability of our proposed functional grasp synthesis model, we input the objects of test set into the semantic segmentation network to obtain the predicted functional areas and 'touch code'. Then the prediction results of the segmentation network are fed into the trained grasp synthesis model to generate the final grasp results. As shown in Fig. 10, from left to right, each image shows the object function area we annotated, the object function area predicted by the segmentation network, and the final grasp result generated by the above process. From the first three lines, it can be found that the grasp generation network can still synthesize good functional grasps for objects with good segmentation results, while from the last line, it can be concluded that the quality of the segmentation results directly determines whether the functional grasp can be generated.

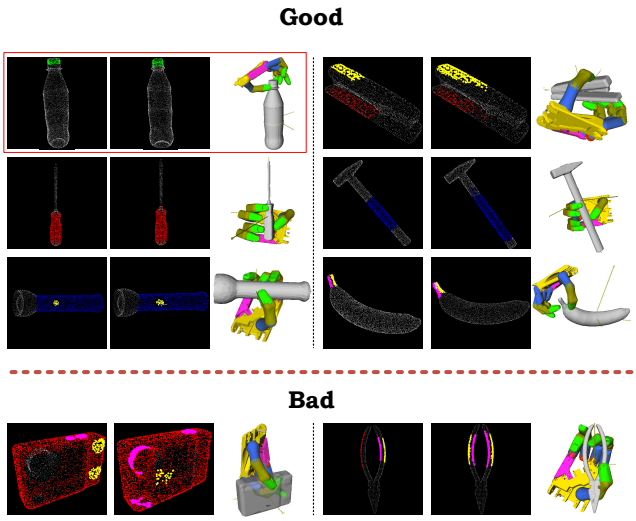


Figure 10: The quality results of functional grasp. Images in the 1st column show the semantic segmentation labels of the objects in test set, while that in the 2nd and 3rd illustrate the predicted semantic segmentation results and grasps generated from the results respectively.

5. Conclusion and Future Work

To make robots manipulate objects like human beings, we accurately segment the functional area of the object for grasping, and annotate touch code for each functional area to guide the dexterous hand to complete the functional grasp and post-grasp task. The rich experimental results show that our proposed grasp synthesis framework can teach dexterous hands to achieve functional grasp. For some of the existing problems, we have also described them in detail, and will solve them by enriching the annotation of the dataset or proposing new constraint function in future work. Additionally, we also plan to use annotated 'intention code' to guide the dexterous hand to perform the post-grasp action for completing manipulation task in future work, so as to truly enable the robot to manipulate objects like a human.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China (No. U1708263, 61873046).

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016. 5

[2] Keni Bernardin, Koichi Ogawara, Katsushi Ikeuchi, and Ruediger Dillmann. A sensor fusion approach for recognizing continuous human grasping sequences using hidden markov models. *IEEE Transactions on Robotics*, 21(1):47–57, 2005. 2

[3] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000. 2

[4] Samarth Brahmbhatt, Cusuh Ham, Charles C. Kemp, and James Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[5] Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. Contactgrasp: Functional multi-finger grasp synthesis from contact. *arXiv preprint arXiv:1904.03754*, 2019. 1, 2, 4

[6] Samarth Brahmbhatt, Chengcheng Tang, Christopher D Twigg, Charles C Kemp, and James Hays. Contactpose: A dataset of grasps with object contact and hand pose. *arXiv preprint arXiv:2007.09545*, 2020. 1, 2, 4

[7] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015. 3

[8] Umberto Castiello. The neuroscience of grasping. *Nature Reviews Neuroscience*, 6(9):726–736, 2005. 3

[9] Enric Corona, Guillem Alenya, Antonio Gabas, and Carme Torras. Active garment recognition and target grasping point detection using deep learning. *Pattern Recognition*, 74:629–641, 2018. 2

[10] Enric Corona, Albert Pumarola, Guillem Alenya, Francesc Moreno-Noguer, and Grégory Rogez. Ganhand: Predicting human grasp affordances in multi-object scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5031–5041, 2020. 1, 2

[11] Hao Dang and Peter K Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1311–1317. IEEE, 2012. 2

[12] Renaud Detry, Jeremie Papon, and Larry Matthies. Task-oriented grasping with semantic and geometric scene understanding. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3266–3273. IEEE, 2017. 2

[13] Staffan Ekvall and Danica Kragic. Learning and evaluation of the approach vector for automatic grasp generation and planning. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4715–4720. IEEE, 2007. 2

[14] Chiara Gabellieri, Franco Angelini, Visar Arapi, Alessandro Palleschi, Manuel G Catalano, Giorgio Grioli, Lucia Pallottino, Antonio Bicchi, Matteo Bianchi, and Manolo Garabini. Grasp it like a pro: Grasp of unknown objects with robotic hands based on skilled human expertise. *IEEE Robotics and Automation Letters*, 5(2):2808–2815, 2020. 2

[15] Oliver Glauser, Shihao Wu, Daniele Panozzo, Otmar Hilliges, and Olga Sorkine-Hornung. Interactive hand pose estimation using a stretch-sensing soft glove. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 2

[16] Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3786–3793. IEEE, 2016. 1, 2

[17] Henning Hamer, Juergen Gall, Thibaut Weise, and Luc Van Gool. An object-dependent hand pose prior from sparse training data. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 671–678. IEEE, 2010. 1

[18] Henning Hamer, Konrad Schindler, Esther Koller-Meier, and Luc Van Gool. Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1475–1482. IEEE, 2009. 1

[19] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3196–3206, 2020. 1

[20] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020. 2

[21] Alexander Herzog, Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, Tamim Asfour, and Stefan Schaal. Template-based learning of grasp selection. In *2012 IEEE International Conference on Robotics and Automation*, pages 2379–2384. IEEE, 2012. 2

[22] Martin Hjelm, Carl Henrik Ek, Renaud Detry, and Danica Kragic. Learning human priors for task-constrained grasping. In *International Conference on Computer Vision Systems*, pages 207–217. Springer, 2015. 2

[23] Divye Jain, Andrew Li, Shivam Singhal, Aravind Rajeswaran, Vikash Kumar, and Emanuel Todorov. Learning deep visuomotor policies for dexterous hand manipulation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3636–3643. IEEE, 2019. 1, 2

[24] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311. IEEE, 2015. 3

[25] Alexander Kasper, Zhixing Xue, and Rüdiger Dillmann. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934, 2012. 3

[26] Mia Kokic, Danica Kragic, and Jeannette Bohg. Learning task-oriented grasping from human activity datasets. *IEEE Robotics and Automation Letters*, 5(2):3352–3359, 2020. 2

[27] Marek S Kopicki, Dominik Belter, and Jeremy L Wyatt. Learning better generative models for dexterous, single-view grasping of novel objects. *The International Journal of Robotics Research*, 38(10-11):1246–1267, 2019. 2

[28] Robert Krug, Dimitar Dimitrov, Krzysztof Charusta, and Boyko Iliev. On the efficient computation of independent contact regions for force closure grasps. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 586–591. IEEE, 2010. 1

[29] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. 2

[30] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018. 2

[31] Hui Li, Jindong Tan, and Hongsheng He. Magichand: Context-aware dexterous grasping using an anthropomorphic robotic hand. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9895–9901. IEEE, 2020. 2

[32] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Generating grasp poses for a high-dof gripper using neural networks. *arXiv preprint arXiv:1903.00425*, 2019. 2, 4

[33] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Deep differentiable grasp planner for high-dof grippers. *arXiv preprint arXiv:2002.01530*, 2020. 2, 4

[34] Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network. In *Robotics Research*, pages 455–472. Springer, 2020. 1

[35] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017. 2

[36] Priyanka Mandikal and Kristen Grauman. Dexterous robotic grasping with object-centric visual affordances. *arXiv preprint arXiv:2009.01439*, 2020. 1, 2, 4

[37] Andrew T Miller and Peter K Allen. Graspit!: A versatile simulator for grasp analysis. In *in Proc. of the ASME Dynamic Systems and Control Division*. Citeseer, 2000. 1, 2, 4, 7

[38] Anh Nguyen, Dimitrios Kanoulas, Darwin G Caldwell, and Nikos G Tsagarakis. Detecting object affordances with convolutional neural networks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2765–2770. IEEE, 2016. 2

[39] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16, 1988. 1

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 5

[41] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016. 2

[42] Domenico Prattichizzo, Monica Malvezzi, Marco Gabiccini, and Antonio Bicchi. On the manipulability ellipsoids of underactuated robotic hands with compliance. *Robotics and Autonomous Systems*, 60(3):337–346, 2012. 1

[43] Markus Przybylski, Tamim Asfour, and Rüdiger Dillmann. Planning grasps for robotic hands using a novel object representation based on the medial axis transform. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1781–1788. IEEE, 2011. 1

[44] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. *arXiv preprint arXiv:2004.04650*, 2020. 1, 2

[45] 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. *arXiv preprint arXiv:1709.10087*, 2017. 1, 2

[46] Maximo A Roa, Max J Argus, Daniel Leidner, Christoph Borst, and Gerd Hirzinger. Power grasp planning for anthropomorphic robot hands. In *2012 IEEE International Conference on Robotics and Automation*, pages 563–569. IEEE, 2012. 1

[47] Máximo A Roa and Raúl Suárez. Computation of independent contact regions for grasping 3-d objects. *IEEE Transactions on Robotics*, 25(4):839–850, 2009. 1

[48] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. From caging to grasping. *The International Journal of Robotics Research*, 31(7):886–900, 2012. 1

[49] Javier Romero, Hedvig Kjellstrom, and Danica Kragic. Modeling and evaluation of human-to-robot mapping of grasps. In *2009 International Conference on Advanced Robotics*, pages 1–6. IEEE, 2009. 2

[50] Carlos Rosales, Raúl Suárez, Marco Gabiccini, and Antonio Bicchi. On the synthesis of feasible and prehensile robotic grasps. In *2012 IEEE International Conference on Robotics and Automation*, pages 550–556. IEEE, 2012. 1

[51] Jungwon Seo, Soonkyum Kim, and Vijay Kumar. Planar, bimanual, whole-arm grasping. In *2012 IEEE International Conference on Robotics and Automation*, pages 3271–3277. IEEE, 2012. 1

[52] Lin Shao, Fabio Ferreira, Mikael Jorda, Varun Nambiar, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Oussama Khatib, and Jeannette Bohg. Unigrasp: Learning a unified model to grasp with multifingered robotic hands. *IEEE Robotics and Automation Letters*, 5(2):2286–2293, 2020. 2

[53] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018. 2

[54] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019. 2

[55] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 509–516. IEEE, 2014. 3

[56] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019. 4

[57] Nikolaus Vahrenkamp, Leonard Westkamp, Natsuki Yamanobe, Eren E Aksoy, and Tamim Asfour. Part-based grasp planning for familiar objects. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 919–925. IEEE, 2016. 2

[58] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8639–8648, 2018. 5

[59] Yuting Ye and C Karen Liu. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012. 1

[60] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019. 1, 2