Refer and Grasp: Vision-Language Guided Continuous Dexterous Grasping

Yayu Huang[1,2,*], Dongxuan Fan[1,2,*], Wen Qi[1,2], Daheng Li[1], Yifan Yang[1,2], Yongkang Luo[1], Jia Sun[1], Qian Liu[1] and Peng Wang[1,2,3,†]

***Abstract*— Robotic grasping guided by natural language instructions faces challenges due to ambiguities in object descriptions and the need to interpret complex spatial context. Existing visual grounding methods often rely on datasets that fail to capture these complexities, particularly when object categories are vague or undefined. To address these challenges, we make three key contributions. First, we present an automated dataset generation engine for visual grounding in tabletop grasping, combining procedural scene synthesis with template-based referring expression generation, requiring no manual labeling. Second, we introduce the RefGrasp dataset, featuring diverse indoor environments and linguistically challenging expressions for robotic grasping tasks. Third, we propose a visually grounded dexterous grasping framework with continuous grasp generation, validated through extensive real-world robotic experiments. Our work offers a novel approach for language-guided robotic manipulation, providing both a challenging dataset and an effective grasping framework for real-world applications. Project website: https://refer-and-grasp.github.io.**

I. Introduction

Robotic grasping plays a crucial role in real-world applications such as manufacturing, healthcare, and household services. The most intuitive way for humans to interact with robots is through natural language, where instructions like "grasp the blue stuff behind the bottle" require robots to parse referring expressions involving object attributes (e.g., color, shape) and spatial relationships. While such expressions are natural for humans, robots operating in unstructured environments often struggle to disambiguate targets in cluttered scenes, particularly when object categories are unknown or descriptions are intentionally vague.

To bridge the gap between linguistic instructions and robotic grasp execution, numerous methods [1]–[4] have been developed based on deep learning visual grounding (VG) frameworks. These approaches generally fall into two categories. Two-stage methods, exemplified by [1], localize the target object through bounding boxes or segmentation masks and generate grasp poses. In contrast, one-stage methods like [2] directly predict grasp poses without explicit object detection.

Despite these advancements, existing methods face several limitations. First, current datasets inadequately support language-guided robotic grasping in unstructured real-world environments. Models trained on widely used VG datasets such as RefCOCO [5] are unsuited for robotic applications, as these datasets focus on outdoor or generic indoor scenes populated with non-graspable objects. Although recent efforts have created robot-centric VG datasets [1]–[3] featuring referring expressions for tabletop objects, many annotations in these datasets rely excessively on redundant categorical information (e.g., explicit object names). This enables models to bypass comprehensive linguistic reasoning by exploiting superficial textual mentions rather than learning meaningful attribute-relationship correlations, therefore limiting their generalization to novel objects absent from the training data.

To address these challenges, we present RefGrasp, a novel synthetic dataset with its generation engine. It is a visual grounding dataset for unstructured tabletop grasping, featuring procedurally generated photorealistic scenes and diverse referring expressions. The dataset specifically emphasizes linguistically complex descriptions through template-generated relationship-centric annotations (e.g., color-based comparisons, projective spatial relations) while incorporating vague referents that omit object names, reflecting natural language usage patterns. Furthermore, we propose a visually grounded dexterous robotic grasping framework supporting continuous grasp generation. The framework consists of a Transformer-based visual grounding network to produce precise target object segmentation masks, and a Conditional Variational Autoencoder (CVAE) [6] with Gated Recurrent Units (GRUs) [7] to generate continuous grasp trajectories through temporal sequence modeling. Comprehensive real-world robotic experiments validate the framework's robustness and operational effectiveness.

In summary, this paper makes three primary contributions:

- A dataset generation engine for visual grounding in unstructured tabletop grasping, combining procedural scene synthesis with template-based referring expression generation that requires no manual labeling.
- The RefGrasp dataset, featuring diverse indoor environments and linguistically challenging expressions specifically designed for robotic grasping applications.
- A visually grounded dexterous grasping framework with continuous trajectory generation, validated through ex-

[1]State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China.

[3]Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science and Innovation, Chinese Academy of Sciences, Hong Kong.

[*]Contributed equally.

[†]Corresponding author: Peng Wang (email: peng_wang@ia.ac.cn).

This work was supported in part by the National Natural Science Foundation of China under Grants (91748131, 62006229, 61771471, and 62303454), in part by the Beijing Natural Science Foundation Haidian Original Innovation Joint Fund Project (L232035, L222154), in part by the Strategic Priority Research Program of Chinese Academy of Science under Grant XDB32050106, and in part by the InnoHK Project.

tensive real-world robot experiments.

II. Related Works

A. Visual Grounding

Visual grounding (VG) is a collective term for Referring Expression Segmentation (RES) and Referring Expression Comprehension (REC). The input to visual grounding consists of an image and a referring expression, while the output is either a bounding box (for REC) or a segmentation mask (for RES), which identifies the object referred to by the expression. Early approaches primarily relied on Convolutional Neural Networks (CNNs) for image feature extraction and Long Short-Term Memory (LSTM) networks for textual feature extraction, followed by multi-modal fusion modules and task-specific heads to generate bounding boxes or segmentation masks [8], [9]. Recent advancements have shifted toward Transformer-based architectures for both image and text feature extraction, combined with diverse modality fusion techniques [10], [11]. With the emergence of the Segment Anything Model (SAM) [12] and the rapid progress in large language models (LLMs), researchers have adapted these models for visual grounding tasks by modifying their architectures and fine-tuning with layer freezing strategies [13], [14]. In this paper, we build our visual grounding network upon PolyFormer [15], a RES model that generates sequences of polygon vertices in an autoregressive manner, extending its capabilities for visual grounding in robotic grasping scenarios.

B. Referring Expression Segmentation Datasets

Existing referring expression segmentation (RES) datasets primarily focus on outdoor scenes, featuring non-graspable objects such as people, animals, and vehicles. Notable examples include RefCOCO [5], RefCOCO+ [5] and RefCOCOg [16]. For indoor and grasping scenarios, several datasets have been developed. OCID-Ref [17], derived from the OCID dataset [18], includes occluded scenes with referring expressions generated through templates and manual annotations, further enriched via scene graph parsing. Similarly, RoboRefIt [1] and OVGrasping [3] employ template-based and manual annotation methods tailored for grasping tasks. OCID-VLG [2], built on the CLEVR engine [19], extends OCID-Grasp [20] by providing segmentation masks and grasp annotations for end-to-end vision-language grasping models. In this paper, the proposed RefGrasp dataset focuses on indoor grasping scenarios and adopts a synthetic approach to enhance object and expression diversity.

C. Dexterous Grasp Generation

Dexterous grasping enables robotic hands to manipulate objects with human-like dexterity, offering significant advantages in flexibility and precision compared to traditional parallel grippers. Analytical methods, such as GraspIt! [21], [22], solve constrained optimization problems based on physical and geometric assumptions to generate stable grasps. Optimization-based approaches, such as differentiable force closure estimators [23], improve grasp stability and diversity. Recent advancements, including penetration energy computation [24] and position-based dynamics [25], have enhanced efficiency. Despite the improvements, these methods often face challenges in computational time and adaptability to novel objects. To address these limitations, data-driven methods have emerged as the mainstream approach in dexterous grasping research. These methods primarily leverage imitation learning [26]–[28], reinforcement learning [29], [30], or a combination of both [31], [32]. However, most existing works focus on generating static grasps for simple single-object scenarios, limiting their applicability to complex, real-world tasks. To bridge this gap, we propose a conditional generative model capable of end-to-end generation of natural and human-like grasping sequences.

III. RefGrasp Dataset

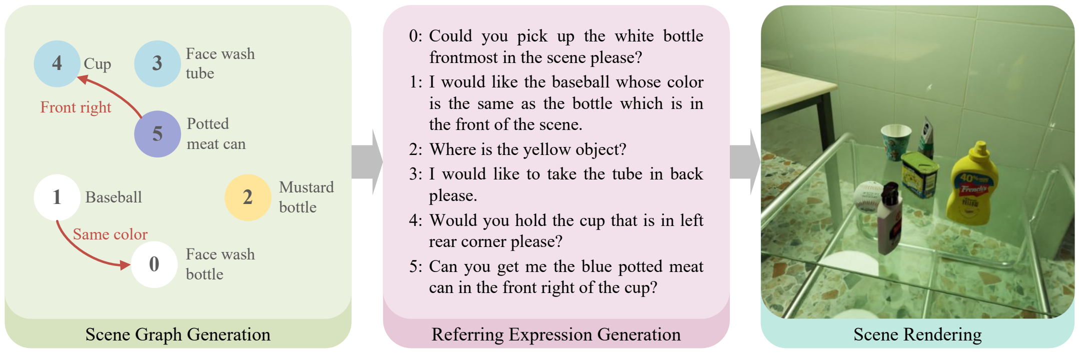
To enable better language-driven grasping perception, we introduce RefGrasp, a synthetic RES dataset specifically designed for unstructured tabletop grasping scenarios, utilizing the physically based rendering pipeline of Blender to generate photo-realistic images. The dataset comprises 36,192 RGB-D images and 36,223 referring expressions in total, describing 91 distinct objects. It is partitioned into training, validation, and test splits, containing 32,205 / 1,989 / 1,998 images and 32,254 / 1,989 / 1,980 expressions, respectively. The dataset is generated using a custom-built dataset generation engine, which can be easily extended to include a broader variety of objects and expression templates, thereby enabling the creation of more complex datasets.

The dataset generation engine consists of a **scene graph generation process**, a **referring expression generation method**, and a **scene rendering pipeline**, as illustrated in Fig. 1. For an empty scene, we first procedurally generate its scene graph by randomly sampling nodes (objects) and edges (relationships between objects). Subsequently, referring expressions are constructed based on the scene graph using a variety of language templates. Next, object locations are determined according to the scene graph, with random selection of surface types, materials for floors and walls, lighting conditions, etc. Finally, the RGB-D images and segmentation masks are rendered using Blender's Cycles rendering engine.

A. Scene Graph Generation

The definition of *scene graph* in this paper is similar to that in [33]. We represent objects and their attributes as nodes, and relationships between objects as edges. Many datasets first randomly sample the types and positions of objects, then annotate the scene using a scene graph [2], [17], [19]. However, we employ a scene graph generation method to construct the scene before placing objects. This approach eliminates the need for post-processing steps (e.g., rejection sampling) to balance the dataset. Moreover, we can easily generate relationships such as multiple same objects in a row, which is challenging when objects are sampled randomly.

To build a scene graph, we construct multiple independent scene trees. In each tree, nodes and edges are generated by



0: Could you pick up the white bottle frontmost in the scene please?
1: I would like the baseball whose color is the same as the bottle which is in the front of the scene.
2: Where is the yellow object?
3: I would like to take the tube in back please.
4: Would you hold the cup that is in left rear corner please?
5: Can you get me the blue potted meat can in the front right of the cup?

Fig. 1. Dataset generation process of RefGrasp. First, a scene graph is generated with objects as nodes and relationships between objects as edges. Next, referring expressions are generated with various templates. Finally, the scene is rendered according to the scene graph.

TABLE I
EXAMPLES OF REFERENCE COMPONENTS APPLIED IN REFGRASP DATASET.

Category	Subcategory	Component Name Example	Referring Expression Example
Location	Relevant to single	Left; front; near	Pick up the bottle **near** the blue box please.
	Relevant to multiple	Middle; row order	Pick up the bottle in the **middle** of the box and cup please.
	Absolute	Frontmost; rear rightmost	Pick up the bottle at **frontmost** please.
	Unspecified	Unspecified	Pick up the bottle please.
Attribute	Direct	Color	The **red** bowl at rear.
	Indirect	Not color; same color	The bowl that has the **same color** as the cup at rear.
	Unspecified	Unspecified	The bowl at rear.
Reference	Name	Name	Please grasp the **coca cola** at rightmost.
	Category	Category	Please grasp the **tin can** at rightmost.
	Vague	Vague	Please grasp the **stuff** at rightmost.

sampling the *locations*, *attributes*, and *references of objects*, which together form *reference components*, are later used to generate referring expressions for the objects. Some of these components are inspired by [2], [34], and examples are shown in Table I. Among these components, we manually annotate the color, name, and category of the objects. Unlike previous datasets, we argue that vague object references (e.g., "stuff" or "thing") are important both in real-world communication and model training, as they reduce redundant information in referring expressions and present a great challenge to the language comprehension ability of a model.

During the generation process, each object is assigned *row* and *column* attributes, similar to pieces on a chessboard, to avoid ambiguity in positioning. Thus, spatial relationships between objects can be represented using two integers. For example, if we represent "in front" as $(+1, 0)$, then "to the left behind" corresponds to $(-1, -1)$. When generating new nodes, we ensure that each referring expression uniquely identifies an object by checking whether existing nodes in the scene already satisfy the new reference components, and by verifying that the new node does not compromise the uniqueness of existing nodes' reference components.

B. Referring Expression Generation

We generate referring expressions for the objects in the scene after constructing the scene graph. Referring expressions are generated using the reference components of the nodes and language templates. Each reference component has several different expressions, from which we sample and construct simple expressions. The structure can be represented as `[<ATTR>+]<OBJ>+<LOC>[+<RE`$_P$`>]` or `<OBJ>[+<ATTR>+<RE`$_P$`>]`, where `<ATTR>` `<LOC>` `<OBJ>` are reference components of the node and `<RE`$_P$`>` is the referring expression of its parent node(s).

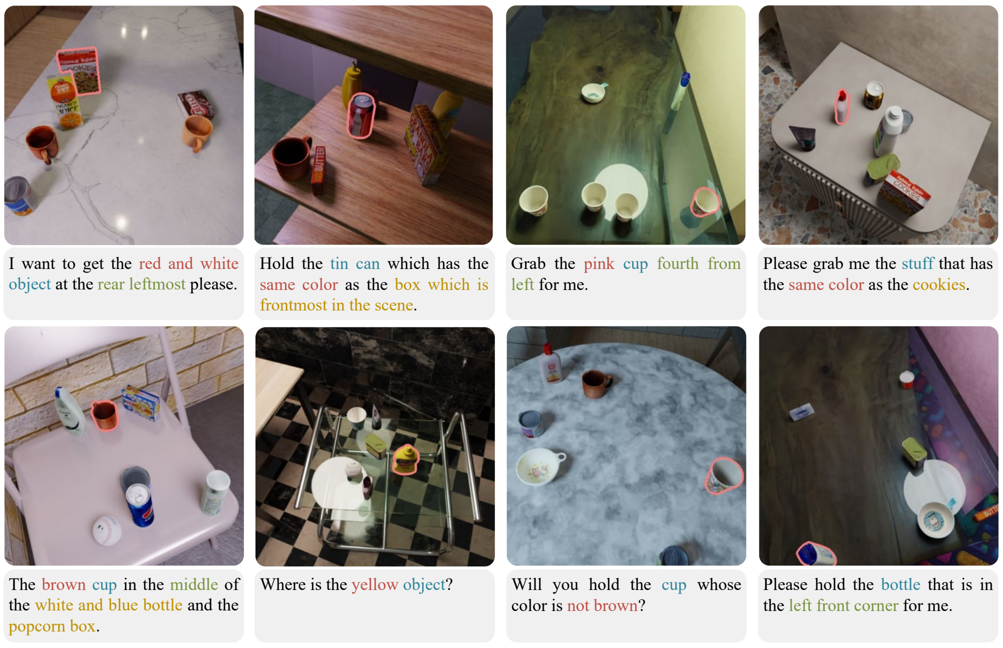
These expressions are integrated into templates to form complete referring expressions for grasping scenarios. The templates include eight sentence structures, such as direct object descriptions (e.g., "The cookie box second from left."), interrogatives (e.g., "Which one is the cookie box second from left?"), and imperatives (e.g., "Please pick up the cookie box second from left for me."), etc.

C. Scene Rendering Pipeline

After generating the scene graph and corresponding referring expressions, we place the object, surface, and background models in Blender and render the scene. We use the Blender API and BlenderProc [35] to establish a dataset rendering pipeline. To enhance realism and randomness, we collected 16 wall textures, 21 floor textures for the background, and 44 surface types of varying sizes for placing objects. All 3D models and materials are open-source. For the graspable objects, we compiled a set of 91 high-quality 3D models from [36]–[38].

The pipeline begins by placing the objects on a randomly selected surface. The position of an object is determined by its row and column in the scene graph with a random offset, while ensuring no collisions between objects. Next, we sample the camera pose, field of view, the lighting condition and background materials of the scene. For each scene graph, the positions of the objects and the surface on which they are placed are sampled only once, while the other aforementioned randomization options are resampled for each rendered image.

Occlusion in the images is also crucial for model training. We aim to include objects that are partially occluded in



I want to get the red and white object at the rear leftmost please.

Hold the tin can which has the same color as the box which is frontmost in the scene.

Grab the pink cup fourth from left for me.

Please grab me the stuff that has the same color as the cookies.

The brown cup in the middle of the white and blue bottle and the popcorn box.

Where is the yellow object?

Will you hold the cup whose color is not brown?

Please hold the bottle that is in the left front corner for me.

Fig. 2. Annotated examples from RefGrasp dataset. In the expressions, object references are marked with blue, attributes are marked with red, locations are marked with green and related objects are marked with yellow.

TABLE II
COMPARISON OF MAIN CHARACTERISTICS BETWEEN PREVIOUS RES/REC DATASETS AND REFGRASP.

Dataset	For Grasp	Data Format	Number of Object Categories	Number of Images	Number of Expressions	Average Length of Expressions
RefCOCO [5]	No	RGB	80	19.9k	142.2k	3.61
Sun-Spot [39]	No	RGB-D	578	1.9k	8.0k	14.04
Cops-Ref [34]	No	RGB	508	703	148.7k	14.4
CLEVR-Ref+ [40]	No	RGB	-	85k	850k	22.38
OCID-Ref [17]	Yes	RGB+3D	58	2.3k	305k	8.56
OCID-VLG [2]	Yes	RGB-D+3D	31	1.7k	89.6k	7.7
RoboRefIt [1]	Yes	RGB-D	66	10.8k	50.7k	9.5
OVGrasping [3]	Yes	RGB	117	23.6k	63.4k	10.0
RefGrasp (ours)	Yes	RGB-D	91	36.2k	36.2k	12.8

some images, but not heavily occluded. Therefore, after sampling the camera pose, we check the occlusion rate of each object. If any object is occluded by more than 60%, the camera pose is resampled. Finally, we use Blender's Cycles rendering engine to render the scene's RGB images, along with instance segmentation masks and depth maps. Examples from the dataset are shown in Fig. 2.

D. Comparison with Existing Datasets

The main characteristics of RefGrasp are shown in Table II. Our dataset distinguishes itself from existing RES or REC datasets in the following ways:

- **The first photo-realistic synthetic dataset focused on visual grounding in grasping scenarios.** While [40] is a synthetic RES dataset, it is intended for diagnostic purposes and not for training models applicable to real-world scenarios. Synthetic scenes in [3] are created in a simplified simulated environment with plain backgrounds, basic lighting conditions, and coarse rendering.
- **A diverse range of graspable objects.** Datasets like [5], [34], [39] include mostly outdoor or indoor non-graspable objects, making them unsuitable for training models focused on robotic grasping, particularly dexterous grasping. Compared to grasp-oriented RES datasets [1], [2], [17], our dataset features a wide variety of graspable objects, making it easily adaptable to real-world applications.
- **More complex referring expressions.** In [1], researchers manually created scenes with ambiguous samples, but reasoning can often be performed from partial descriptions. In [2], attributes and relations are added only when the reference is unclear, excluding expressions with attributes or relations alone. In RefGrasp, we generate unstructured scenes and include vague object references in some expressions, requiring models to reason based solely on location, attributes, or object relationships.

IV. VISION-LANGUAGE GUIDED CONTINUOUS DEXTEROUS GRASPING

A. Overview

The architecture of the proposed system is illustrated in Fig. 3. The system is designed as a two-stage pipeline, with each stage involving a separate network trained individually.

Stage I focuses on perception and segmentation tasks. The input to the network includes an RGB image, a depth image, and natural language instructions, and the output is the segmentation mask of the target object. **Stage II** addresses the reasoning and decision-making aspects of grasping. First, the depth image, segmentation mask, and robot's current state are fed into the grasp generation network. The output is the grasp configuration, which includes a sequence of actions for the robot to perform the grasp. The detailed processes of each stage are described below.

B. Visual Grounding Network

In grasping scenarios, depth information provides critical spatial relationships and edge features to enhance image segmentation accuracy. Building upon the PolyFormer [15] architecture, we augment the model by incorporating a depth channel input. We use a depth encoder to extract depth-specific features, which are flattened into a sequential representation: $F_D \in \mathbb{R}^{(H/32 \times W/32) \times C_D}$, where H and W denote the original height and width of the input. These depth features are then aligned with RGB image features (F_V) and linguistic features (F_L) through linear projection layers into a unified multimodal feature space. The fused multimodal features are formulated as:

$$F_M = [f_{\text{V2M}}(F_V),\ f_{\text{D2M}}(F_D),\ f_{\text{L2M}}(F_L)] \tag{1}$$

where $f_{\text{V2M}}(\cdot)$, $f_{\text{D2M}}(\cdot)$, and $f_{\text{L2M}}(\cdot)$ represent learnable linear mappings. Here, $F_V \in \mathbb{R}^{(H/32 \times W/32) \times C_V}$ and $F_L \in \mathbb{R}^{L \times C_L}$ denote the RGB image features and text features respectively. The resulting multimodal feature F_M has a dimensionality of $F_M \in \mathbb{R}^{(2(H/32 \times W/32)+L) \times C_M}$, where L corresponds to the sequence length of the text embeddings. This fused representation is processed by a Transformer encoder for feature integration and subsequently fed into the decoder. Consistent with PolyFormer's design, the decoder employs an autoregressive strategy to iteratively predict the points of the target object's bounding box and polygon vertices of the segmentation mask.

C. Grasp Generation

The grasp generation model uses a CVAE [6], guided by conditional input from a re-coded scene point cloud. This point cloud, derived from the visual grounding network's binary mask and depth image, has a fourth channel where target object points are set to 1 and all other points to 0.

The point cloud encoder first processes the scene point cloud $\mathcal{P}^S \in \mathbb{R}^{N_0 \times 4}$, which is passed through a PointNet encoder to extract features. The future motion sequence $\{h_t\}, t \in \{1, 2, \ldots, T\}$ is then encoded by the robot state encoder, which is implemented as fully-connected layers. The resulting embeddings are then fed into a GRU [7] encoder to capture the temporal dependencies across the frames in the sequence. These features are then concatenated and passed through two fully connected layers. The final output consists of the mean μ and the variance σ^2, which parameterize the posterior Gaussian distribution $Q(z|\mu, \sigma^2)$. The GRU decoder receives the latent variable z and the

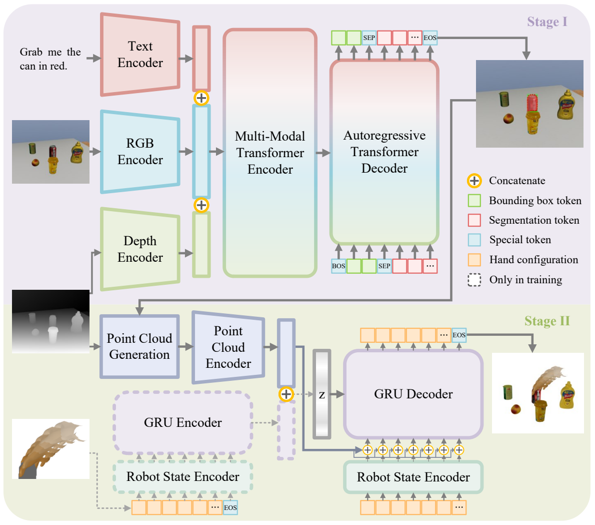


Fig. 3. The framework of our method. Given an RGB-D image and a text instruction, the visual grounding network outputs the segmentation mask and the continuous grasp generation network generates a grasp trajectory.

encoded scene features. The robot's current state start is fed as the first input to the GRU, which is initialized with z and autoregressively generates the future motion sequence $\{h'_t\}, t \in \{1, 2, \ldots, T\}$. The output sequence is passed through fully connected layers to reconstruct the robot's future motion sequence. Additionally, a stop token is generated at each time step using the sigmoid function to indicate sequence completion.

The first objective function is the reconstruction error, which measures the difference between the predicted and true future robot poses. We use L1 loss to compute this error, denoted as $\mathcal{L}_{\mathcal{R}} = \frac{1}{T}\sum_{t=1}^{T} |h'_t - h_t|$, where T represents the number of frames. To improve contact accuracy, we define a contact loss $\mathcal{L}_{\mathcal{C}}$, which measures the distance between the final hand contact points V^C and the target object point cloud $\mathcal{P}^T$, calculated as $\mathcal{L}_{\mathcal{C}} = \frac{1}{N}\sum_i \mathcal{D}(V_i^C, \mathcal{P}^T)$, where N represents the number of hand contact points. We also define a termination loss, which evaluates the accuracy of the stop token generated at each time step. This loss is computed using binary cross-entropy to predict whether the sequence has completed, denoted as $\mathcal{L}_{\mathcal{T}} = -\frac{1}{T}\sum_{t=1}^{T} (y_t \log(\hat{y}_t) + (1 - y_t)\log(1 - \hat{y}_t))$, where y_t is the ground truth stop token at time step t, and $\hat{y}_t$ is the predicted stop token. Finally, following the training of VAE, we define a KL loss to ensure that the latent variable distribution $Q(z|\mu, \sigma^2)$ is close to a standard Gaussian distribution, which is achieved by minimizing the KL divergence, i.e., $\mathcal{L}_{\mathcal{KL}} = \text{KL}(Q(z|\mu, \sigma^2)||\mathcal{N}(0, I))$. The final loss function is given by:

$$\mathcal{L} = \lambda_{\mathcal{R}}\mathcal{L}_{\mathcal{R}} + \lambda_{\mathcal{C}}\mathcal{L}_{\mathcal{C}} + \lambda_{\mathcal{T}}\mathcal{L}_{\mathcal{T}} + \lambda_{\mathcal{KL}}\mathcal{L}_{\mathcal{KL}} \tag{2}$$

where $\lambda_{\mathcal{R}}$, $\lambda_{\mathcal{C}}$, $\lambda_{\mathcal{T}}$ and $\lambda_{\mathcal{KL}}$ are weighting factors for the respective terms.

V. Experiments

A. Visual Grounding Experiments

To validate the effectiveness of our architectural enhancements, we conduct comprehensive evaluations on the RefGrasp dataset, with additional analysis of model performance and robustness across referring expression categories.

1) Implementation: Our visual grounding model integrates a depth feature encoder based on ConvNeXt-Tiny [41], modified to accept single-channel normalized depth images as input. The remaining architecture follows PolyFormer-B [15], which employs Swin-B [42] as the RGB image encoder and BERT-base [43] as the text encoder. Both the Transformer encoder and decoder consist of 6 layers.

2) Training Details: We initialize model weights with PolyFormer's checkpoint on RefCOCOg [16]. The first training phase involves joint training on RefGrasp and RoboRefIt [1] for 15 epochs with full parameter updates, using RGB-text pairs only. In the second phase, we freeze the RGB and text encoders, introduce depth inputs, and conduct 15 epochs of dataset-specific training on RefGrasp. All training phases use a batch size of 192 with the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$), weight decay of 1×10^{-8}, and an initial learning rate of 5×10^{-5} with polynomial decay. Input images are resized to 512×512.

3) Results: Table III compares visual grounding performance on RefGrasp, evaluating both bounding box accuracy (AP50) and segmentation quality (mIoU/oIoU). Our model achieves consistent improvements over the RGB-only PolyFormer model (+0.08% AP50, +0.28% mIoU, +0.42% oIoU), confirming the utility of depth integration. Furthermore, we assess the performance of pretrained models, including PolyFormer [15] and LISA++-7B [44] (denoted by †). The experimental findings reveal that existing visual grounding methods, while effective in general domains, exhibit limited transferability to robotic grasping scenarios without domain-specific fine-tuning.

To further investigate model's robustness for different language components, we evaluate performance across referring expression categories (Table IV). The results demonstrate that our model achieves robust performance across most categories. Specifically, the model exhibits strong spatial reasoning with 97.52-98.34% AP50 for location-based references and precise attribute recognition (98.81% AP50 for direct attributes). However, performance declines for indirect attributes (94.75% AP50) and drops significantly for vague references (91.18% AP50), revealing limitations in implicit relationship reasoning. These findings underscore the importance of future research to address semantic ambiguity and contextual reasoning challenges, particularly for vague or under-specified instructions, which are common in real-world robotic applications.

TABLE III
Comparison of visual grounding results on RefGrasp.

Method	Input Format	AP50	mIoU	oIoU
PolyFormer† [15]	RGB+Text	28.95	26.92	10.40
LISA++-7B† [44]	RGB+Text	26.38	35.47	29.67
PolyFormer [15]	RGB+Text	96.10	81.18	78.67
Ours	RGB-D+Text	**96.18**	**81.46**	**79.09**

TABLE IV
Model performance on different reference components.

Description Type	AP50	mIoU	oIoU
Location-Single	97.52	81.63	79.77
Location-Multiple	97.53	**83.68**	**83.33**
Location-Absolute	98.34	82.47	81.72
Attribute-Direct	**98.81**	82.94	82.32
Attribute-Indirect	94.75	83.16	80.77
Reference-Vague	91.18	77.47	72.00

B. Real-Robot Experiments

Real-robot experiments were conducted to demonstrate the performance and practicality of the proposed framework. The experimental setup included a 6-DoF Elfin robotic arm equipped with a CASIA Speed Hand with 1 DoA and 10 DoF, and a 3D Kinect camera mounted overhead for scene perception. The retargeting and scoring methods followed the approach described in our previous work [45]. For each experiment, 200 samples were collected, and the sample with the highest score was selected for execution.

1) Training Details: We extract human demonstrations from the DexYCB dataset [46], which includes 1000 cluttered tabletop scenarios with 20 YCB objects [36]. Each sequence shows a human grasping a specific object from the cluttered scene, with hand postures represented by the MANO hand model. To augment the dataset, we apply random transformations, including rotations and plane shifts. The model is trained using a batch size of 288 with an initial learning rate of 5×10^{-4} and polynomial decay, using default Adam optimizer. The training process runs for a total of 300 epochs. The loss weights were set as $\lambda_{\mathcal{R}} = 1$, $\lambda_{\mathcal{C}} = 0.05$, $\lambda_{\mathcal{T}} = 0.01$ and $\lambda_{\mathcal{KL}} = 1$.

2) Comparative Results on Object Categories: To assess the generalization ability of the proposed framework, we evaluated its performance on both familiar object categories with novel instances and completely unseen object categories, as presented in Table V. The system achieved strong results across all test cases, with success rates of 100% for the Milk Box, 80% for the Magic Cube and the Cookie Box, and 70% for the Apple when handling new instances of trained categories. For novel object categories not present during training, the framework maintained strong performance, achieving 90% on the Chip Bucket and 70% on the Sponge, indicating effective adaptation to previously unseen geometries and material properties. Performance slightly dropped for the Bread and Tea Can, both at 60%. The lower success rate on the Bread was primarily due to frequent collisions with the tabletop during grasp execution, while the reflective surface of the Tea Can caused point cloud distortions that led to grasp planning errors. These results demonstrate the framework's ability to generalize across

TABLE V
Grasp Success Rates for Seen and Unseen Categories

Object	Seen	Unseen	Success Rate
Magic cube	✓		8/10
Apple	✓		7/10
Milk Box	✓		10/10
Cookie Box	✓		8/10
Sponge		✓	7/10
Bread		✓	6/10
Chip Bucket		✓	9/10
Tea Can		✓	6/10

TABLE VI

GRASP SUCCESS RATES FOR DIFFERENT REFERENCE COMPONENTS

Description Type	Milk Box				Juice Box				Sponge				Chip Bucket			
	Perc	Plan	Act	Overall	Perc	Plan	Act	Overall	Perc	Plan	Act	Overall	Perc	Plan	Act	Overall
Location-Single	20/20	20/20	19/20	95%	19/20	19/20	19/20	95%	19/20	18/20	15/20	75%	19/20	19/20	18/20	90%
Location-Multiple	19/20	19/20	19/20	95%	20/20	20/20	19/20	95%	18/20	17/20	15/20	75%	19/20	19/20	18/20	90%
Location-Absolute	20/20	20/20	19/20	95%	20/20	20/20	18/20	90%	19/20	19/20	16/20	80%	20/20	20/20	19/20	95%
Attribute-Direct	20/20	20/20	19/20	95%	20/20	20/20	19/20	95%	19/20	18/20	15/20	75%	20/20	20/20	19/20	95%
Attribute-Indirect	19/20	19/20	19/20	95%	19/20	19/20	19/20	95%	18/20	17/20	16/20	80%	18/20	18/20	17/20	85%
Reference-Vague	18/20	18/20	18/20	90%	19/20	19/20	19/20	95%	16/20	15/20	14/20	70%	18/20	18/20	17/20	85%

both category-level and instance-level variations, while also revealing practical limitations when dealing with challenging object properties such as low-profile geometries or reflective materials.

3) Comparative Results on Different Reference Components: To evaluate the performance systematically in real-robot experiments, we divide the process into three stages: Perception (Perc), Planning (Plan), and Action (Act), with success rates recorded separately for each stage, as shown in Table VI and visualized in Fig. 4. Among all types, `Location-Absolute` and `Attribute-Direct` yield the most robust and consistent perception performance, achieving near-perfect success across all objects. For rigid objects such as the Milk Box, Juice Box, and Chip Bucket, these references contribute to consistently high planning and action success rates, typically exceeding 90%, with only occasional execution errors. In contrast, objects with more complex properties, like the Sponge, exhibit somewhat lower overall success, primarily due to increased challenges during planning and execution stages. `Location-Single` and `Location-Multiple` also demonstrate high overall success rates, while `Attribute-Indirect` results in moderately reduced perception and planning performance due to increased ambiguity in language grounding. `Reference-Vague` descriptions produce the lowest and least consistent success across all objects, reflecting the difficulty in resolving underspecified instructions. Overall, these results underscore the importance of precise and unambiguous references for successful grasping, while also revealing that object properties can still lead to occasional failures despite accurate perception.

VI. CONCLUSION

In conclusion, this paper tackles key challenges in robotic grasping using natural language instructions, focusing on ambiguities in object descriptions and spatial relationships. We introduce an automated engine for generating visual grounding datasets, combining procedural scene synthesis with template-based referring expression generation, thereby eliminating the need for manual labeling. The RefGrasp dataset, tailored for robotic grasping, provides diverse indoor environments and complex linguistic expressions. Furthermore, we present a visually grounded dexterous grasping framework with continuous grasp generation, demonstrating its robustness through extensive real-world experiments. In the future, we will continue to explore new human-robot interaction paradigms, focusing on enhancing adaptability and precision in dynamic environments.

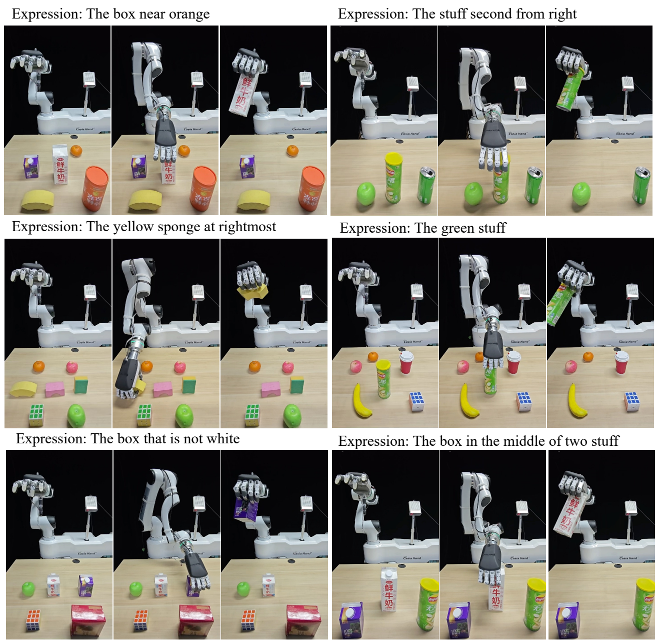

Expression: The box near orange
Expression: The stuff second from right
Expression: The yellow sponge at rightmost
Expression: The green stuff
Expression: The box that is not white
Expression: The box in the middle of two stuff

Fig. 4. Visualized results for different reference components.

REFERENCES

[1] Y. Lu, Y. Fan, B. Deng, F. Liu, Y. Li, and S. Wang, "Vl-grasp: a 6-dof interactive grasp policy for language-oriented objects in cluttered indoor scenes," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 976–983.

[2] G. Tziafas, X. Yucheng, A. Goel, M. Kasaei, Z. Li, and H. Kasaei, "Language-guided robot grasping: Clip-based referring grasp synthesis in clutter," in *7th Annual Conference on Robot Learning*, 2023.

[3] M. Li, Q. Zhao, S. Lyu, C. Wang, Y. Ma, G. Cheng, and C. Yang, "Ovgnet: A unified visual-linguistic framework for open-vocabulary robotic grasping," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 7507–7513.

[4] J. Li, K. Ye, H. Kang, M. Liang, Y. Wu, Z. Liu, H. Zhuang, R. Huang, and Y. Chen, "Grasp what you want: Embodied dexterous grasping system driven by your voice," 2024. [Online]. Available: https://arxiv.org/abs/2412.10694

[5] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, "Modeling Context in Referring Expressions," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science, 2016, pp. 69–85.

[6] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[7] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Oct. 2014, pp. 103–111.

[8] R. Hu, M. Rohrbach, and T. Darrell, "Segmentation from Natural Language Expressions," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science, 2016, pp. 108–124.

[9] L. Yu, Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg, "MAttNet: Modular Attention Network for Referring Expression Comprehension," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1307–1315.

[10] M. Li and L. Sigal, "Referring Transformer: A One-step Approach

to Multi-task Visual Grounding," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 19 652–19 664.

[11] Z. Yang, J. Wang, Y. Tang, K. Chen, H. Zhao, and P. H. Torr, "LAVT: Language-Aware Vision Transformer for Referring Image Segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 18 134–18 144.

[12] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[13] Y. Zhang, T. Cheng, R. Hu, L. Liu, H. Liu, L. Ran, X. Chen, W. Liu, and X. Wang, "EVF-SAM: Early Vision-Language Fusion for Text-Prompted Segment Anything Model," Aug. 2024, arXiv:2406.20076 [cs]. [Online]. Available: http://arxiv.org/abs/2406.20076

[14] X. Lai, Z. Tian, Y. Chen, Y. Li, Y. Yuan, S. Liu, and J. Jia, "Lisa: Reasoning segmentation via large language model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9579–9589.

[15] J. Liu, H. Ding, Z. Cai, Y. Zhang, R. K. Satzoda, V. Mahadevan, and R. Manmatha, "PolyFormer: Referring Image Segmentation as Sequential Polygon Generation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 18 653–18 663.

[16] J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy, "Generation and Comprehension of Unambiguous Object Descriptions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 11–20.

[17] K.-J. Wang, Y.-H. Liu, H.-T. Su, J.-W. Wang, Y.-S. Wang, W. Hsu, and W.-C. Chen, "OCID-ref: A 3D robotic dataset with embodied language for clutter scene grounding," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Jun. 2021, pp. 5333–5338.

[18] M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "EasyLabel: A Semi-Automatic Pixel-wise Object Annotation Tool for Creating Robotic RGB-D Datasets," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6678–6684.

[19] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 1988–1997.

[20] S. Ainetter and F. Fraundorfer, "End-to-end Trainable Deep Neural Network for Robotic Grasp Detection and Semantic Segmentation from RGB," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 13 452–13 458.

[21] S. Brahmbhatt, C. Tang, C. D. Twigg, C. C. Kemp, and J. Hays, "Contactpose: A dataset of grasps with object contact and hand pose," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, 2020, pp. 361–378.

[22] 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.

[23] 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.

[24] 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," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 359–11 366.

[25] D. Turpin, T. Zhong, S. Zhang, G. Zhu, E. Heiden, M. Macklin, S. Tsogkas, S. Dickinson, and A. Garg, "Fast-grasp'd: Dexterous multi-finger grasp generation through differentiable simulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8082–8089.

[26] 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.

[27] D. Hidalgo-Carvajal, H. Chen, G. C. Bettelani, J. Jung, M. Zavaglia, L. Busse, A. Naceri, S. Leutenegger, and S. Haddadin, "Anthropomorphic grasping with neural object shape completion," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8034–8041, 2023.

[28] W. Wei, P. Wang, S. Wang, Y. Luo, W. Li, D. Li, Y. Huang, and H. Duan, "Learning human-like functional grasping for multifinger hands from few demonstrations," *IEEE Transactions on Robotics*, vol. 40, pp. 3897–3916, 2024.

[29] B. Wu, I. Akinola, A. Gupta, F. Xu, J. Varley, D. Watkins-Valls, and P. K. Allen, "Generative attention learning: a "general" framework for high-performance multi-fingered grasping in clutter," *Autonomous Robots*, vol. 44, no. 6, pp. 971–990, 2020.

[30] T. Wu, M. Wu, J. Zhang, Y. Gan, and H. Dong, "Learning score-based grasping primitive for human-assisting dexterous grasping," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[31] F. Ficuciello, A. Migliozzi, G. Laudante, P. Falco, and B. Siciliano, "Vision-based grasp learning of an anthropomorphic hand-arm system in a synergy-based control framework," *Science robotics*, vol. 4, no. 26, p. eaao4900, 2019.

[32] L. Huang, W. Cai, Z. Zhu, and Z. Zou, "Dexterous manipulation of construction tools using anthropomorphic robotic hand," *Automation in Construction*, vol. 156, p. 105133, 2023.

[33] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3668–3678.

[34] Z. Chen, P. Wang, L. Ma, K.-Y. K. Wong, and Q. Wu, "Cops-Ref: A New Dataset and Task on Compositional Referring Expression Comprehension," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 10 083–10 092.

[35] M. Denninger, D. Winkelbauer, M. Sundermeyer, W. Boerdijk, M. Knauer, K. H. Strobl, M. Humt, and R. Triebel, "Blenderproc2: A procedural pipeline for photorealistic rendering," *Journal of Open Source Software*, vol. 8, no. 82, p. 4901, 2023.

[36] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, Sep. 2015.

[37] S. Tyree, J. Tremblay, T. To, J. Cheng, T. Mosier, J. Smith, and S. Birchfield, "6-DoF Pose Estimation of Household Objects for Robotic Manipulation: An Accessible Dataset and Benchmark," in *International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 081–13 088.

[38] H.-S. Fang, M. Gou, C. Wang, and C. Lu, "Robust grasping across diverse sensor qualities: The GraspNet-1Billion dataset," *The International Journal of Robotics Research*, vol. 42, no. 12, pp. 1094–1103, 2023.

[39] C. Mauceri, M. Palmer, and C. Heckman, "Sun-spot: An rgb-d dataset with spatial referring expressions," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 1883–1886.

[40] R. Liu, C. Liu, Y. Bai, and A. L. Yuille, "CLEVR-ref+: Diagnosing visual reasoning with referring expressions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4180–4189.

[41] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 966–11 976, 2022.

[42] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9992–10 002.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, Jun. 2019, pp. 4171–4186.

[44] S. Yang, T. Qu, X. Lai, Z. Tian, B. Peng, S. Liu, and J. Jia, "Lisa++: An improved baseline for reasoning segmentation with large language model," *arXiv preprint arXiv:2312.17240*, 2023.

[45] Y. Huang, D. Fan, D. Yan, W. Qi, G. Deng, Z. Shao, Y. Luo, D. Li, Z. Wang, Q. Liu, and P. Wang, "Human-robot collaborative telegrasping in clutter with five-fingered robotic hands," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2215–2222, 2025.

[46] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal, S. Birchfield *et al.*, "Dexycb: A benchmark for capturing hand grasping of objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9044–9053.