A Joint Network for Grasp Detection Conditioned on Natural Language Commands

Yiye Chen[1], Ruinian Xu[1], Yunzhi Lin[1], and Patricio A. Vela[1]

Abstract— We consider the task of grasping a target object based on a natural language command query. Previous work primarily focused on localizing the object given the query, which requires a separate grasp detection module to grasp it. The cascaded application of two pipelines incurs errors in overlapping multi-object cases due to ambiguity in the individual outputs. This work proposes a model named *Command Grasping Network* (*CGNet*) to directly output command satisficing grasps from RGB image and textual command inputs. A dataset with ground truth (image, command, grasps) tuple is generated based on the VMRD dataset to train the proposed network. Experimental results on the generated test set show that *CGNet* outperforms a cascaded object-retrieval and grasp detection baseline by a large margin. Three physical experiments demonstrate the functionality and performance of *CGNet*.

I. INTRODUCTION

The ability to understand natural language instructions, written or spoken, is a desirable skill for an intelligent robot, as it allows non-experts to communicate their demands. For manipulation, grasping the target is an indispensable first step in fulfilling the request. This paper addresses the problem of object grasp recognition based on a natural language query. Two aspects are included in the problem: *what* is the target object required by the command, and *how* to grasp it. A substantial body of research focuses on the first aspect [1]–[7], termed *natural language object retrieval*, but few consider explicitly and/or jointly the second.

A naive solution for the second is to first locate the target then plan the grasp in 3D space [8]. An alternative is to first use task-agnostic multiple grasp detection, then select a grasp coinciding with the object using a suitable heuristic. Both are prone to error in visual clutter due to imprecise target localization or the inability disambiguate grasps when objects overlap (see §IV for examples).

In this paper, we delay the explicit object retrieval process and propose a *natural language grasp retrieval* solution that directly detects grasps based on command understanding as depicted by the input/output structure of Fig. 1. We argue that doing so is more efficient compared to cascaded solutions and reduces the aforementioned errors because it removes the burden of accurate object segmentation, and is less influenced by distractor objects as they are rarely included in graspable regions as shown in the output grasp boxes of Fig. 1. Our hypothesis is that *the retrieval and understanding of the*

target object can be implicitly done within a deep network, such that semantic and object information is encoded into the grasp feature representation. Confirming the hypothesis involves designing a *natural language grasp retrieval* deep network to output grasps conditioned on the input command. The network outputs the grasp box location and the probability distribution over the classification space of discretized orientations, retrieval state, and background state. Two garbage classes induce decision competition to reject unsuitable candidates. Training the network requires the construction of a multi-objects dataset with associated command and grasp annotations, generated by semi-automatically augmenting the VMRD dataset [9] using a templates-parsing approach with paraphrase augmentation for the command set. Evaluating trained *CGNet* performance on a vision dataset and via physical experiments validates the hypothesis.

A. Related Work

1) Natural Langauge Object Retrieval: Natural Language Object Retrieval addresses the problem of locating a target object in an image when specified by a natural language instruction. The main challenge lies in interpreting visual and textual input to capture their correspondence. Early works parsed perceptual and textual input with human-designed rules, which have less expressive embeddings [10], [11].

The advent of deep learning provided tools to extract high level semantic meaning from both image and language signals. Text-based image retrieval research adopted convolutional neural networks (CNNs) and recurrent neural networks (RNNs) as image and text encoders [2]. Inspired by the success of object detection, more recent work focuses on aligning a specific object to the text by segmenting the image and encoding each object region [1], [4], [5], [12]. Segmentation modules used include external region proposal algorithms [4], [12], object detectors [1], or built-in object region proposal network [5]. Work in [3] also investigated incorporating both local and global information for instance-level queries in the text. The correspondence between visual and textual input can then be obtained by grounding-via-generation strategy [3], [4], [12], or designing a proximity-based similarity score function in the shared image-and-text embedding space such as inner-product [1], [12] or cosine similarity [6]. Instead of segmenting object-scale regions, our work aims to retrieve grasp regions. Though such regions are smaller in scale and contain object parts, our work demonstrates that sufficient object information is encoded in the grasp region representations to reason with commands and differentiate objects.

[1] Y. Chen, R. Xu, Y. Lin, and P.A. Vela are with the School of Electrical and Computer Engineering, and the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA. {yychen2019, rnx94, ylin466, pvela}@gatech.edu
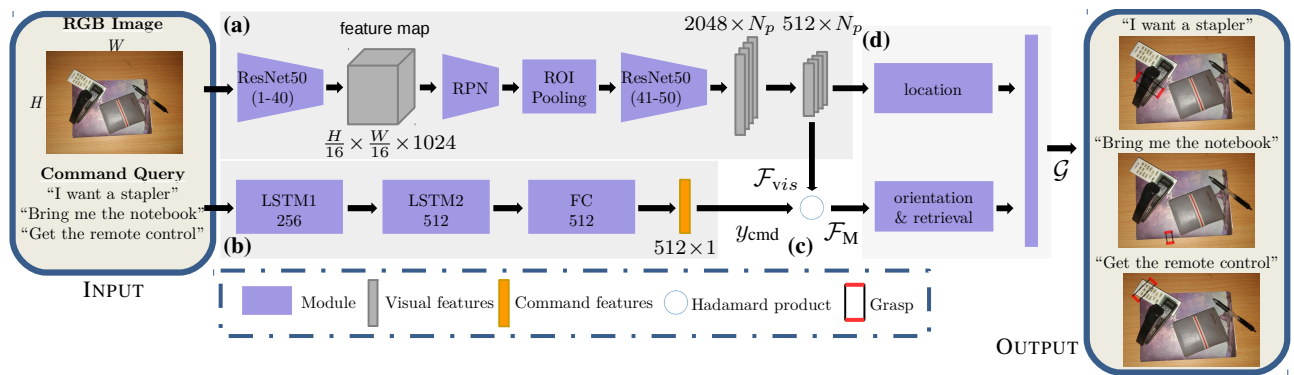
Fig. 1. Depiction of the deep network processing structure. Given an image and a text command: **(a)** multiple visual grasp features are extracted; **(b)** the command is decoded into a command feature; **(c)** the visual and command outputs are merged; **(d)** and then processed to identify applicable grasps.

More specific topics studied in robotics include addressing object attributes specification in commands such as shape [13], spatial relations [8], or functionality [6]. Sometimes ambiguous commands need clarification through human-robot dialogue [7], [8]. Again, a main difference is the desire to bypass the object retrieval stage and influence grasp predictions from commands. The research is complementary.

2) Task-agnostic Grasp Detection: Task-agnostic grasp detection aims to detect all graspable locations from the perceptual input. While classical approaches are based on mechanical constrains [14], [15], recent studies resorted to CNN to capture the geometric information to either predict quality scores for sampled grasps [16], to directly output the grasp configurations [17]–[20], or to capture collision information [21]. We leverage this research to design a command-influenced grasp detection network, by augmenting a 2-stage grasp detection network [22] whose architecture consists of grasp region proposals followed by grasp prediction.

3) Semantic Grasp Detection: Semantic grasp detection seeks functionally suitable grasps for specific downstream tasks [23]–[26], such as the identification of different grasps on an object for different purpose. More closely related to our task is that of grasping of a specific target in a cluttered environment [9], [27]–[29], where shared encoding of object and grasp content is shown to be important. Moving to task-agnostic grasp detection increases the sequential complexity of the solution for object retrieval [27], [28]. As noted earlier, the problem at hand also needs to address textual comprehension from the input command and its alignment with the grasp regions. While [9] investigated reasoning over object-scale regions, our hypothesis allows omission of the object classification in image-text feature fusion.

B. Problem Statement

Given an RGB image I and a corresponding natural language command C (e.g., *"Give me the banana"*) requesting an object, the *command-based grasp selection* problem is to generate a set of grasp configurations $\mathcal{G}$ capable of grasping and retrieving the object if it is present in the image. It requires establishing a function f such that $\mathcal{G} = f(I, C)$. The envisioned end-effector is a gripper with two parallel plates, such that grasping is executed with a vertically downward grasp approach direction relative to a horizontal surface.

Outputting grasps as a 5D vector, $g = [x, y, \theta, w, h]^T$, defined with respect to the 2D image I is sufficient to plan such a grasping action. The 5D vector g describes the region in the image plane that the manipulator gripper should project to prior to closing the parallel plates (or equivalent) on the object. The coordinates (x, y, θ) are the grasp center and orientation (relative to the world z-axis), and (w, h) are the width and height of the pre-closure grasp region.

The scene may contain a single or multiple objects. The target object can be partially occluded and the function f should still provide a grasp if enough of the object is visible and can be grasped. It should also determine whether a detected grasp is suitable or not from the input command.

II. PROPOSED APPROACH

The network architecture for grasps recognition based on visual and text query inputs is depicted in Figure 1. Inspired by the success of feature merging for joint image-and-text reasoning problems [5], [30], [31], we integrate natural language command interpretation with grasp detection via merged visual and textual representations. To facilitate feature merging, a two-stage grasp detection model provides the base network structure [22] so that the merger occurs in the second stage. Consequently, the pipeline in Figure 1 depicts two independent and parallel processes first: an image feature extractor to get a set of visual grasp-and-object sensitive features from the input RGB image, and a language feature extractor to get the command feature representation. The image feature extractor relies on a region proposal network to identify a set of potential graspable regions from which the visual features are obtained. The final stage fuses both feature spaces and predicts the *5D parameters* of candidate grasps plus the *suitability* of the grasps for the given command. This section covers the details of the network structure.

A. Grasp Region Feature Extraction

The first stage of the visual processing pipeline proposes a set of grasp regions of interest (ROIs) [9], [22], and their feature representations, which are expected to contain not only geometric information for determining grasp configuration, but also semantic information for reasoning with commands. At the end of the grasp proposal pipeline [22] the fixed-size feature maps are passed to a shallow convolutional network

to produce a set of vectors in $\mathbb{R}^{D_I}$, which are interpreted as the embedding for each candidate grasp region. The output is set of such vectors,

$$\mathcal{F}_{\mathrm{vis}} = \{y_{\mathrm{vis},i}\}_1^{N_P} = \{(\rho^i, r^i, y_{\mathrm{img}}^i)\}_1^{N_P} = f_{\mathrm{vis}}(I),$$

where the coordinates of $(\rho, r, y_{\mathrm{img}} \in \mathbb{R}^{D_I})$ consist of the proposal probability, the predicted position, and the grasp region feature representation.

A natural approach would be to sequentially apply object detection then grasp recognition in a *detect-then-grasp* pipeline, where the object of interest comes from the command interpreter. Sequential use with independent training does not exploit a deep network's ability to learn joint representations. Moving to a *detect-to-grasp* paradigm [9] creates multiple branches after ROI pooling for learning joint representations similar to the *detect-to-caption* pipeline of [5]. In our case the *object* category is not given, nor visually derived, but must be decoded from the command. Since the object category is still unknown, it should not have primacy. The decision should be delayed. To give grasps primacy, the ROI detector and feature space primarily key on grasp regions, not object regions. Filtering grasp candidates by object category occurs when the visual grasp and command feature spaces merge. By virtue of the multi-task objective and joint training, the process of detecting the target object is implicitly accomplished within the network. A benefit is that the grasp ROIs have object-specific attention-like characteristics, which comes from our observation that a grasp coordinate output g generally targets a specific object and rarely includes other objects (see §IV). As a result, it can reduce confusion when there are occlusion cases due to multiple objects in the scene. This deep network design reflects a *grasp-to-detect* paradigm. The results in §IV will demonstrate that it is feasible to encode the object information into these smaller regions, which is critical for reasoning with commands.

B. Command Feature Extraction

Encoding a natural language command into a vector feature involves first mapping the command into a sequence of vectors using a trainable word embedding table. The vector sequence is then passed to a command encoder to map the sequence to a single vector representation.

1) Word Embedding: A word embedding table is a set of vectors $\{e_i \in \mathbb{R}^{D_W}\}_{i=1}^n$ representing n words separately, so that each word $word_i$ is encoded as e_i. The set of n known words and their vector representations is called the *dictionary*. For robustness to out-of-dictionary words, an unknown word token $\langle \mathrm{unk} \rangle$ is appended to the dictionary. The embedding vectors will be learnable so that they can be optimized to the problem described in §I-B.

2) Command Encoder: The encoder is a 2-layer LSTM [32] with hyperbolic tangent ($\tanh$) activation. LSTM networks extract content or knowledge from sequences, which makes them effective for joint vision and language problems [30], [31]. An additional fully connected layer (FC 512) with ReLU activation is added after the second LSTM block

to increase the model's capacity. The LSTM maps each input vector (the embedding of a word in the command) to an output vector that factors in earlier input data due to its feedback connection. After sequentially providing the entire command the last sequentially output vector defines the command feature representation output. The command feature inference is summarized as: $y_{\mathrm{cmd}} = f_{\mathrm{cmd}}(C) \in \mathbb{R}^{D_C}$.

C. Feature Merging and Output Layer

The second stage takes the command vector y_{cmd} and a set of grasp region features $\mathcal{F}_{\mathrm{vis}}$ as input, then predicts grasp configurations and their matching probability for the command query:

$$\mathcal{F}_{\mathrm{grasp}} = \{y_{\mathrm{grasp},i}\}_1^{N_P} = \{(\gamma_i, g_i)\}_1^{N_P} = f_{\mathrm{grasp}}(\mathcal{F}_{vis}, \mathcal{F}_{\mathrm{M}}, y_{\mathrm{cmd}})$$

with (γ, g) the classification probability and grasp vector.

The objective is to merge the grasp candidate information with the command information for conditioning the retrieved grasps in the visual and textual inputs. Prior to merging the two intermediate output streams, a fully-connected layer f_{red} performs dimension reduction of the visual signals in $\mathbb{R}^{D_I}$ to match the command signal in $\mathbb{R}^{D_C}$ when there is dimension mismatch (e.g., $D_I > D_C$). The Hadamard product (point-wise product) then merges the features:

$$y_{\mathrm{M}} = f_{\mathrm{M}}(y_{\mathrm{img}}, y_{\mathrm{cmd}}) = f_{red}(y_{\mathrm{img}}) \odot y_{\mathrm{cmd}},$$

and get feature set $\mathcal{F}_{\mathrm{M}}$, where $\odot$ is the element-wise product.

The final computation involves two sibling, single-layer, fully-connected networks for grasp fitting, with a position regression branch and an orientation classification branch [22]. For the position regression branch (*location* block), the position of a grasp should not be conditioned on the command query. Accordingly, the network receives the image branch outputs $\mathcal{F}_{\mathrm{vis}}$ as inputs, and outputs a 4D position (x, y, w, h) for each orientation class. The orientation classification branch (lower block) includes output classes for rejecting the candidates in $\mathcal{F}_{\mathrm{vis}}$ and is where the command feature influences the outcome. The input is a merged feature y_{M}. The output orientation space consists of N_{orient} classes plus two additional classes for rejecting grasp candidates that are not sensible. The first, from [22], is a *background (BG)* class. The second is a *not_target (NT)* class for feasible grasps that do not reflect the target object associated to the command request. The two classes differ since the *BG* class indicates regions where it is not possible to grasp (e.g., no object should be there). Although a double-stream setup (i.e, singling out the language retrieval score prediction as a separate branch) is also an option [5], [29], combining commands and grasps into the same outcome space (i.e., orientation class) eliminates the need for a retrieval confidence threshold and employs decision competition to determine the preferred outcome, resulting in a set of candidate grasps $\mathcal{G}$.

D. Loss

Though the network structure includes a second command branch that merges with the main grasping branch, the loss function for it does not significantly differ from that of [22].

It consists of proposal $\mathcal{L}_{\text{p}}$ and grasp configuration $\mathcal{L}_{\text{g}}$ losses, $\mathcal{L} = \mathcal{L}_{\text{p}} + \mathcal{L}_{\text{g}}$, to propagate corrections back through both branches during training.

The proposal loss primarily affects the grasp ROI branch, e.g., f_{vis}. A ground truth (GT) binary proposal position and label (for positive ROI) is defined for each ROI (ρ^*, r_i^*) where the GT binary class label is `True` if a grasp is specified, and `False` if not. Both r^* and r are 4-dimensional vectors specifying center location and size: (x, y, w, h). The loss is:

$$\mathcal{L}_{\text{p}}(\{\rho_i, r_i\}_1^{N_p}, \{\rho_i^*, r_i^*\}_1^{N_p})$$
$$= Z_{\text{cls}}^{-1} \sum_{i=1}^{N_{\text{cls}}^p} L_{\text{cls}}(\rho_i, \rho_i^*) + Z_{\text{loc}}^{-1} \sum_{i=1}^{N_p} \rho_i^* L_{\text{loc}}(r_i, r^*),$$

where $Z_{\text{cls}} = N_{\text{cls}}^p$ and $Z_{\text{loc}} = \sum_i^{N_p} \rho_i^*$ are normalization constants, and N_{cls}^p is a hyperparameter specifying the number of ROIs sampled for the loss calculation. The grasp binary label loss L_{cls} is the cross entropy loss, and the grasp location loss L_{loc} is the smooth L1 loss [33].

The term $\mathcal{L}_{\text{g}}$ guides the final grasp detection output $\mathcal{F}_{\text{grasp}}$. GT position and class for each ROI is denoted as r_i^* and c_i^* separately. c^* is assigned in the following way: (1) if the ROI is assigned as negative in the proposal stage, then c^* is set to the *BG* class; (2) if the positive ROI is associated to a non-target ROI, then the c^* is set to *NT* class; (3) otherwise, c^* is set to the corresponding orientation class according to GT orientation angle. The grasp loss is:

$$\mathcal{L}_{\text{g}}(\{(\gamma_i, g_{0:3,i})\}_1^{N_p}, \{c_i^*, r_i^*\}_1^{N_p}) =$$
$$\frac{1}{N_{\text{cls}}^g} \sum_{i=1}^{N_{\text{cls}}^g} L_{\text{cls}}(\gamma_i, c_i^*) + \frac{1}{N_{pos}} \sum_{i=1}^{N_p} \mathbf{1}[c_i^*! = bg] L_{\text{loc}}(g_{0:3,i}, r_i^*),$$

where N_{cls}^g is a hyperparameter; $N_{pos} = \sum_{i=1}^{N_p} \mathbf{1}[c_i^*! = bg]$; $\mathbf{1}[*]$ is the indicator function that is one when the condition is satisfied and zero otherwise.

III. METHODOLOGY

This section covers the network configuration and training process for instantiating *CGNet*. Training will require an annotated dataset compatible with the network's input-output structure, whose construction is described here. Since the network will be evaluated as an individual perception module and integrated into a manipulation pipeline, this section also details the experiments and their evaluation criteria.

A. Network Structure and Parameters

The visual processing backbone network is ResNet-50. The base layer and the first three blocks of ResNet-50 are used as the image encoder, while the last block and the final global average pooling layer is used to extract vector representations after ROI-pooling. The visual features have dimension $D_I = 2048$. For the command processing pipeline, the word embedding dimension is $D_E = 128$. The output dimension for the LSTM layers are set to $y_{LSTM1} \in \mathbb{R}^{256}$, $y_{LSTM2} \in \mathbb{R}^{512}$. The dimension of the textual command feature is set to $D_C = 512$. The grasp orientation uses $N_{\text{orient}} = 19$ classes. (w/180° symmetry).

B. Annotated Training Data

Training the proposed network requires a dataset with tuples $(I, C, \mathcal{G})$, where the grasp configurations in $\mathcal{G}$ are associated with the command C. Not aware of such a dataset, we created one by applying template-based command generation to the multi-objects VMRD dataset [9]. The VMRD dataset provides labelled objects and grasps, $(I, \mathcal{O}, \mathcal{G})$, where each grasp $g \in \mathcal{G}$ is associated with an object $o \in \mathcal{O}$ present in I. We convert the object label to a natural language command by parsing a randomly chosen template, such as *"Pass me the $\langle obj \rangle$"*, from a pre-defined set [6]. This generation is limited to commands with object categories explicitly stated, and the object class must be in VMRD. The network can learn more free-formed commands, like those demanding a function instead of an object [6].

1) Command Augmentation: The 11 templates adopted from [6] only include *subject-verb-object* and *verb-object*. To enrich the vocabulary size and syntactic diversity, we augment the template set based on the automatic paraphraser. We first append 7 templates with different grammatical structures (*e.g. "Fetch that $\langle obj \rangle$ for me"*). Then a group of commands is generated using the initial template sets and VMRD object labels, one paraphrase is obtained for each command using the paraphraser, meaningless paraphrases are filtered out manually, and finally the new template sets are acquired by removing the object label in the paraphrases. The above step is repeated 10 times, with 35 commands generated for each. The paraphraser used is QuillBot. In the end, 123 templates are generated, that differ from the initial set, such as *"Grab the $\langle obj \rangle$ and bring it to me"* and *"The $\langle obj \rangle$ is the one I must have"*.

2) VMRD+C Dataset: The base VMRD dataset consists of 4683 images with over 10^5 grasps, and approximately 17000 objects labelled into 31 categories. It is split into 4233 and 450 images for training and testing set respectively. For each object in a source image, an $(I, \mathcal{O}, \mathcal{G})$ tuple is generated with the strategy described above. Ground truth orientation classes for grasps associated to the target object are set according to their angle, with the rest are labelled *NT*. Furthermore, command requests with target objects not present in the image are added to the data set for each image. The command is generated from a random object excluding the ground truth ones present in the image, with all grasps set to *NT*. The strategy results in 17387(12972 have-target and 4415 no-target) training data and 1852 (1381 have-target and 471 no-target) testing instances.

C. Training And Testing

1) Training Details: Merging all ROI vision features with textual features slows down training. Applying feature fusion for only positive regions [5] is inconsistent with the inference procedure, where background features will also be merged since no ground truth label is available. Instead, we sample an equal number of negative ROIs as positive ones for feature fusion. The rest are retained for training the visual feature vector output of f_{vis}. This strategy improves convergence.

4579

The initial network is ResNet-50 pretrained on ImageNet, with the other layers and word embeddings randomly initialized. Training for the ⟨unk⟩ token, applies random word dropout with 0.1 probability. The number of ROIs sampled for loss calculations are $N_{cls}^p = N_{cls}^g = 128$. The Adam Optimizer [34] with parameters $(\alpha, \beta) = (0.9, 0.99)$ and initial learning rate of 10^{-4} is used. Training was for $\sim 400k$ iterations with batch-size of 1.

2) Testing Details: During testing, 300 ROIs with top proposal scores are sent to the final grasp prediction layer. After getting the prediction results, The higher value from two garbage classes is used as the threshold to reject candidates independently. Non-maximum suppression over the remaining results leads to the output grasp set $\mathcal{G}$.

D. Physical Experiments

Three physical experiments are designed to demonstrate the effectiveness and the application value of the proposed method in a perception-execution pipeline. The objects used are unseen instances of the known categories.

1) Single Unseen Object Grasping: The unseen single object grasping experiment evaluates the generalization ability of *CGNet*. An unseen instance is randomly placed on the table, and a command requiring an object type is given. For each presented object, we repeated 10 trials for the case where the required and presented objects match, and 5 trials where there is no match (i.e., the requested object is not the presented one). The command is automatically generated using a random template and an input object label. The experiment uses same 8 objects in [9].

2) Multiple Unseen Objects Grasping: The aim is to select and grasp the target from amongst multiple objects, based on the command query. For each trial, a target object with 4 other interfering objects are presented, and the target is demanded by command. Each target is tested for 10 trials, and cover both cases where the target is on top or is partially covered. The target set and the command generation method is the same as in single-object experiment.

3) Voice Command Handover: A human will submit a command verbally to test *CGNet*'s to generalize to unknown words or sentence structures. After the voice command is translated to text using Google Speech Text API, the experiment follows the multi-objects experiment. The robot arm executes the predicted grasp and passes the object to the human operator. The experiment is repeated for 33 trials, with 17 visible targets and 16 partially occluded.

E. Metrics

The perception-only experiments test the correctness of the grasp output, while the manipulation experiments test how well the outputs function in a perceive-plan-act pipeline with an embodied manipulator. The scoring is described below.

1) Perception: We evaluate the model's response to the natural language command query. Scoring adopts the object retrieval **top-k recall(R@k)** and **top-k precision(P@k)** metrics to evaluate multiple grasp detections [3]. R@k is the percentage of cases where at least one of the top-k

Method	R@1	R@3	R@5	R@10	P@1	P@3	P@5	P@10	FPS
Agn-Rnd	25.4	25.1	26.5	24.9	27.0	26.1	25.6	25.6	**9.1**
Ret-Gr	63.1	73.1	76.5	80.8	67.4	68.5	68.3	66.5	4.5
CGNet	**74.9**	**88.2**	**91.0**	**93.2**	76.1	75.3	74.8	72.2	8.7
CG+Ret	74.6	86.8	88.7	90.1	**78.0**	**77.5**	**76.2**	**73.3**	4.4

detections is correct. P@k computes the correct rate for all top-k selections. A correctly detected grasp has a Jaccard Index (intersection over union) greater than 0.25 and an absolute orientation error of less than $30°$ relative to at least one of the ground truth grasps of the target object.

2) Physical Experiment: To evaluate physical experiment, we separate the pipeline into different stages and record their success rate respectively. The three stages considered are: *object retrieval*, *grasp detection*, and *grasp execution*, to identify which step leads to trial failure. The success of the first two are visually confirmed from the bounding box, while the last requires the target to be grasped, picked up, and held for 3 seconds. The voice command handover experiment includes the percentage of the spoken command being exactly translated. The percentage of successful trials, termed *overall* success rate, is also recorded.

IV. RESULTS

This section discusses the evaluation results of both the perception module and the designed physical experiments. Three baselines are compared with our methods:

1) *Random (Agn-Rnd):* A state-of-art task-agnostic grasp detection network MultiGrasp [22] followed by a random choice. The model is re-trained on VMRD.

2) *Cascade (Ret-Gr):* A cascading state-of-art natural language object retrieval model [5] and Multigrasp [22]. The target object is the retrieval region with highest score. Grasps within the retrieval region are kept and ranked based on center-to-center distance (minimum first). Both models were re-trained with VMRD+C.

3) *Obj-Gr:* A grasp detection with object classification network from ROI-based grasp detection [9]. The results are obtained from their published work, which we take as an upper bound since it skips the need to interpret commands.

We also evaluate *CG+Ret*, which takes *CGNet* output and ignores grasps not within the object retrieval region.

A. Vision

Table I lists the visual accuracy and efficiency of the evaluated methods. All methods outperform *Agn-Rnd*. The hypothesized value of encoding semantic information into grasp regions and skipping object detection is evident from the higher values for *CGNet* over *Ret-Gr*, by around 10% on average. Examples presented Fig. 2 demonstrate some of the problem of the cascade baseline. One class of errors is inaccurate target localization, where the natural language object retrieval model yield imprecise locations, which leads to false grasp detection. *CGNet* avoids this step and this type of error. Another class has errors from overlapping or occluding objects, whereby the object boxes include distractors

"I need to buy myself a paper" | "I need to get a toothpaste of my own" | "The towel is the one I need to have" | "For this I need pen" | "I need a notebook"

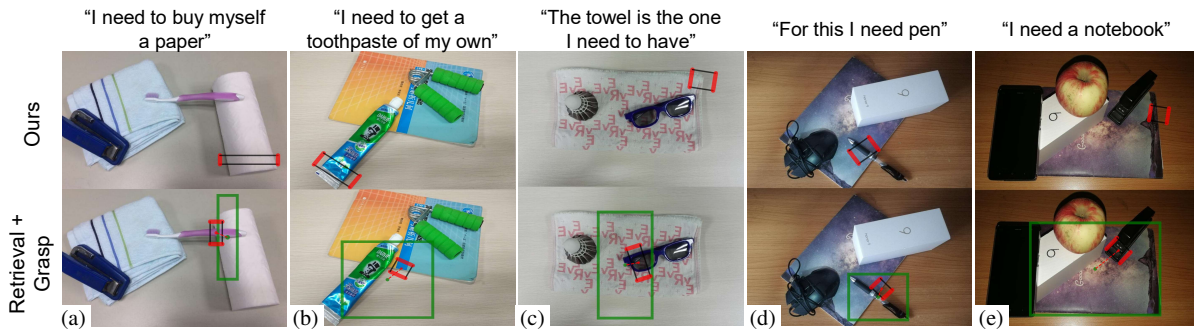Ours | Retrieval + Grasp

(a) (b) (c) (d) (e)

Fig. 2. Grasp detection results for *CGNet* and *Ret-Gr*. The red box denotes the top grasp detection, and the green box object retrieval results. *CGNet* avoids mistakes caused by: **(a)(b)(c)** inaccurate localization of the target object; and **(d)(e)** target bounding boxes with interfering objects.

TABLE II

UNSEEN SINGLE OBJECT GRASPING

Objects	CGNet			Ret-Gr			Obj-Gr[1]		
	Obj	Grp	Exe	Obj	Grp	Exe	Obj	Grasp	Exe
Apple	-	10	9	10	10	8	-	10	9
Banana	-	10	10	10	10	10	-	10	10
Wrist Developer	-	10	9	10	9	9	-	10	10
Tape	-	10	10	10	10	10	-	8	8
Toothpaste	-	10	10	9	9	9	-	10	10
Wrench	-	9	9	5	5	5	-	10	8
Pliers	-	10	8	1	1	1	-	10	10
Screwdriver	-	10	9	5	5	5	-	10	9
Mean	-	9.86	9.25	7.5	7.38	7.13	-	9.75	9.25

[1] Results adopted from original paper.

TABLE III

UNSEEN MULTIPLE OBJECTS GRASPING

Objects	CGNet			Ret-Gr			Obj-Gr[1]		
	Obj	Grp	Exe	Obj	Grp	Exe	Obj	Grasp	Exe
Apple	-	10	8	10	10	10	-	10	9
Banana	-	9	9	9	7	7	-	10	10
Wrist Developer	-	8	8	9	6	6	-	7	6
Tape	-	7	7	9	7	7	-	7	7
Toothpaste	-	8	8	5	4	4	-	8	8
Wrench	-	7	6	5	4	4	-	10	8
Pliers	-	7	7	5	2	2	-	9	9
Screwdriver	-	7	7	3	3	3	-	10	10
Mean	-	7.88	7.50	6.88	5.38	5.38	-	8.88	8.38

[1] Results adopted from original paper.

TABLE IV

VOICE COMMAND HANDOVER

Voice Translation	Grasp Prediction	Execution
19/33	25/33	23/33

objects. Grasp regions are small enough that usualy a single object is attached to them, thereby avoiding confusion from overlapping objects.

CGNet tests on the 471 *NT* data, for which no grasps should be proposed, achieves 65.0% success rate, indicating that sometimes *CGNet* fails to recognize objects at the grasp-level. Applying the object retrieval information as prior helps distinguish between objects, as evidenced by the improved P@K of *CG+Ret* over *CGNet* alone. The trade-off is a lower recall thereby causing a drop in R@K value.

B. Physical Experiments

1) Single Unseen Object Grasping : The generalization ability of *CGNet* is evident in Table II. Though the tested objects were not in the training data the overall detection and execution success rate matches *Obj-Gr* which does not have to perform command interpretation. The *Ret-Gr* baseline is expected to have strong results also, but there is a significant performance drop for the *Wrench*, *Pliers*, and *Screwdriver*. It may result from the domain shift of the unseen objects.

2) Multiple Unseen Object Grasping : Here, all algorithms experience a performance drop as seen in Table III. *CGNet* performs closer to *Obj-Gr* than to *Ret-Gr*, indicating that *CGNet* has learnt to encode similar object-level content in the grasp feature descriptors. However, the reduced value also indicates that object-level discrimination is not as strong as it could be. Some form of nonlocal attention is most likely needed, or loose coupling to object-level feature descriptors.

3) Voice Command Handover: For the voice command version, the comand interpreter does a better job than the voice-to-text network, see Table IV . Out-of-vocabulary words are input to *CGNet* due to translation error ("want"

to "*walked*", "Help" to "*How*") , unexpected descriptions ("... on the *table*"), or colloquial words ("...*please*"). Under these challenges, *CGNet* still extracts the key information and achieves a reasonable 70.0%(23/33) task execution rate. It outperforms *Ret-Gr* on multiple objects and almost matches its single object performance.

All of the outcomes support the value of task prioritization (here grasps) for contextual interpretation of action commands. They also indicate that non-local information provides important information in cases of clutter, where other objects may have similar grasp-level feature encodings.

V. CONCLUSION

This paper presents *Command Grasping Network*(*CGNet*), a network that detects grasp conditioned on text input corresponding to a natural language command. By skipping the object retrieval step and directly detecting grasps, *CGNet* avoids the errors incurred by inaccurate object localization and post-processing of cascaded object retrieval and grasp detection models. Vision dataset evaluation and three proposed physical experiment demonstrate the effectiveness and the generalization ability of *CGNet*.

Future work will explore implicit commands where the object is not in the comand proper but one of its properties is referenced. We also would like to incorporate higher-level or nonlocal visual cues to enhance grasp recognition rates. With both improvements, we envision that the system would be more effective at interacting with a human.

REFERENCES

[1] A. Karpathy, A. Joulin, and L. F. Fei-Fei, "Deep fragment embeddings for bidirectional image sentence mapping," in *Advances in neural information processing systems*, 2014, pp. 1889–1897.

[2] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 207–218, 2014.

[3] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell, "Natural language object retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, "Grounding of textual phrases in images by reconstruction," in *European Conference on Computer Vision*. Springer, 2016, pp. 817–834.

[5] A. Nguyen, Q. D. Tran, T.-T. Do, I. Reid, D. G. Caldwell, and N. G. Tsagarakis, "Object captioning and retrieval with natural language," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[6] T. Nguyen, N. Gopalan, R. Patel, M. Corsaro, E. Pavlick, and S. Tellex, "Robot Object Retrieval with Contextual Natural Language Queries," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[7] J. Hatori, Y. Kikuchi, S. Kobayashi, K. Takahashi, Y. Tsuboi, Y. Unno, W. Ko, and J. Tan, "Interactively picking real-world objects with unconstrained spoken language instructions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3774–3781.

[8] M. Shridhar and D. Hsu, "Interactive visual grounding of referring expressions for human-robot interaction," in *Proceedings of Robotics: Science and Systems*, 2018.

[9] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng, "Roi-based robotic grasp detection for object overlapping scenes," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4768–4775.

[10] J. Krishnamurthy and T. Kollar, "Jointly learning to parse and perceive: Connecting natural language to the physical world," *Transactions of the Association for Computational Linguistics*, vol. 1, pp. 193–206, 2013.

[11] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Francisco, CA, 8 2011, pp. 1507–1514.

[12] S. Guadarrama, E. Rodner, K. Saenko, N. Zhang, R. Farrell, J. Donahue, and T. Darrell, "Open-vocabulary object retrieval." in *Robotics: science and systems*, vol. 2, no. 5, 2014, p. 6.

[13] V. Cohen, B. Burchfiel, T. Nguyen, N. Gopalan, S. Tellex, and G. Konidaris, "Grounding language attributes to objects using bayesian eigenobjects," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1187–1194.

[14] C.-P. Tung and A. C. Kak, "Fast construction of force-closure grasps," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 615–626, 1996.

[15] C. Ferrari and J. F. Canny, "Planning optimal grasps." in *ICRA*, vol. 3, 1992, pp. 2290–2295.

[16] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," 2017.

[17] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[18] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1316–1322.

[19] J. Watson, J. Hughes, and F. Iida, "Real-world, real-time robotic grasping with convolutional neural networks," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2017, pp. 617–626.

[20] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 769–776.

[21] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6232–6238.

[22] F.-J. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018.

[23] H. Dang and P. 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*. IEEE, 2012, pp. 1311–1317.

[24] A. B. Rao, K. Krishnan, and H. He, "Learning robotic grasping strategy based on natural-language object descriptions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 882–887.

[25] W. Liu, A. Daruna, and S. Chernova, "Cage: Context-aware grasping engine," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2550–2556.

[26] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 202–216, 2020.

[27] D. Guo, T. Kong, F. Sun, and H. Liu, "Object discovery and grasp detection with a shared convolutional neural network," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2038–2043.

[28] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg, "Mechanical search: Multi-step retrieval of a target object occluded by clutter," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1614–1621.

[29] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine, "End-to-end learning of semantic grasping," in *Conference on Robot Learning*, 2017, pp. 119–132.

[30] D. Huk Park, L. Anne Hendricks, Z. Akata, A. Rohrbach, B. Schiele, T. Darrell, and M. Rohrbach, "Multimodal explanations: Justifying decisions and pointing to the evidence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8779–8788.

[31] J.-H. Kim, K. W. On, W. Lim, J. Kim, J.-W. Ha, and B.-T. Zhang, "Hadamard Product for Low-rank Bilinear Pooling," in *The 5th International Conference on Learning Representations*, 2017.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.