Equivariant Open-vocabulary Pick and Place via Language Kernels and Patch-level Semantic Maps

Anonymous Author(s)
Affiliation
Address
email

Abstract: Controlling robots through natural language instructions in open-vocabulary scenarios is pivotal for enhancing human-robot collaboration and complex robot behavior synthesis. However, achieving this capability poses significant challenges due to the need for a system that can generalize from limited data to a wide range of tasks and environments. Existing methods rely on large, costly datasets and struggle with generalization. This paper introduces **G**rounded **E**quivariant **M**anipulation (**GEM**), a novel approach that leverages the generative capabilities of pre-trained vision-language models and geometric symmetries to facilitate few-shot and zero-shot learning for open-vocabulary robot manipulation tasks. Our experiments demonstrate GEM's high sample efficiency and superior generalization across diverse pick-and-place tasks in both simulation and real-world experiments, showcasing its ability to adapt to novel instructions and unseen objects with minimal data requirements. GEM advances a significant step forward in the domain of language-conditioned robot control, bridging the gap between semantic understanding and action generation in robotic systems.

Keywords: Language-conditioned Robotic Manipulation, Zero-shot Learning

1 Introduction

Commanding a robotic manipulator with open-vocabulary natural language is important for enabling human-robot collaboration. Taking into account the current state of the environment, the system must map the language instructions onto desired robot actions. The challenge is making systems robustly interpret language about unseen objects and generalize manipulation actions from small amounts of training data.

Existing vision and language models [1, 2, 3] open the possibilities of performing open-vocabulary (or zero-shot) robot manipulation tasks. However, when performing zero-shot in robotic manipulation, existing modular-based approaches [4, 5, 6, 7] do not accurately perform complex, fine-grained manipulation tasks such as "Grasp the mug *by its handle* and put it in the box". Learning-based approaches like CLIPort [8] and RT-2 [9] address this problem by using imitation learning with pre-trained features, where a person demonstrates a task through teleoperation and then learns a manipulation policy from these demonstrations. The challenge here is that robot demonstrations are expensive, and learned policies do not transfer to other manipulation targets in an open-ended way. For example, with CLIPort, a command like "pick the green mug" does not necessarily transfer to a related command such as "pick the red mug" unless a red mug was also present in its training set. As a result, enormous amounts of robot data are required to perform manipulation tasks.

We address this problem by introducing new learning algorithms for imitation learning that leverage large vision and language models to enable generalization, and use equivarient learning to enable efficient learning from small datasets. Our approach, **G**rounded **E**quivariant **M**anipulation (**GEM**) exploits domain symmetries that exist in the robotics aspect of the problem, specifically equivariance

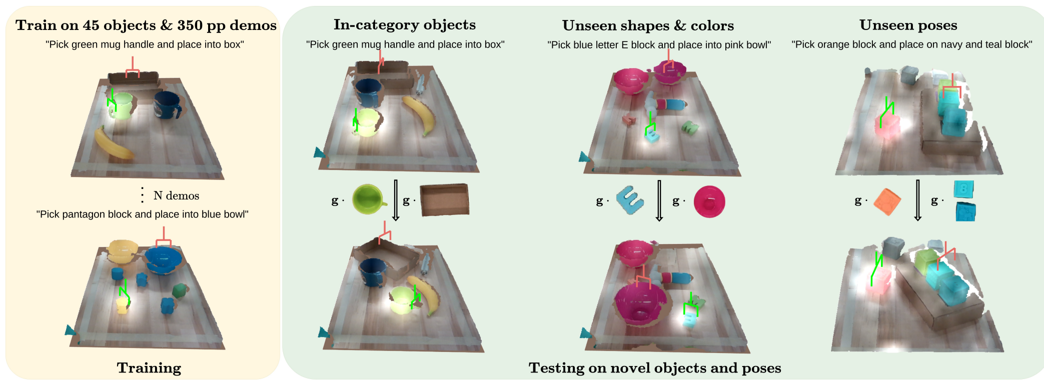


Figure 1: **Overview.** Our method is trained on a small amount of demonstrations (yellow) and can generalize to novel objects and scenes during testing (green). The object being picked is highlighted by the semantic map. In the green testing section, upper and bottom images show spatial generalization when transformation $g \in \mathrm{SE}(2)$ acts on objects. Different columns show zero-shot generalization on unseen colors and shapes given open-vocabulary instructions. The green and red grippers denote pick and place positions.

in $\mathrm{SE}(2)$. For example, consider the task of "grasp the coffee mug by its handle." If there is a rotation or translation of the mug, the desired pick action should also transform accordingly, i.e. equivariantly. If we can incorporate such language-conditioned symmetries into our model, the policy can generalize learned knowledge to many different scenarios related by symmetry transformations automatically, thus making the learning more efficient. Apart from leveraging the symmetries, we show how to incorporate information from large vision and language models to enable few-shot and zero-shot generalization across objects, colors, shapes, and poses, as shown in Figure 1.

Since many complex manipulation tasks can be completed with a sequence of pick-place actions, we frame the learning task as language-conditioned pick and place. We make the following specific contributions. **(1)** We propose a novel method for generating semantic maps for language-conditioned pick-place using large vision and language models. **(2)** We systematically analyze the symmetries underlying language-conditioned pick-place tasks and design language-conditioned steerable adaptive kernels to leverage them. **(3)** We demonstrate the state-of-the-art generalization ability and sample efficiency of our method in simulation, one physical tabletop setting, and one mobile manipulation platform on a series of challenging language-conditioned manipulation tasks. Our evaluation demonstrates that our approach meets or exceeds the few/zero-shot performance of state-of-the-art baselines, while using only 10%-20% training data compared with CLIPort [8] and 0.1% training data compared with VIMA [10] in most of the simulation and real-world tasks.

2 Related work

Language-conditioned policy learning: With the rapid advance in NLP, many recent works attempted to encode language instructions into robot policy learning. [11, 12, 13, 10, 14, 15] use feature concatenation, FiLM [16], or the cross-attention mechanism to fuse image and language features from pre-trained VLM/LLMs. For example, Shridhar et al. [8] utilizes CLIP visual and text encoders and aligned language and image features with a two-stream fusion architecture. [14, 15] process the language token and visual token jointly with transformers [17, 18] for keyframe policy learning. By appending a deep learnable module on pre-trained features, these models are prone to overfit to the training set and lose the generalization ability provided by the pre-trained models. As a result, a copious number of robot data is still required to train these models and the performance largely decreases with unseen objects. For instance, Stepputtis et al. [19] needs 30k datapoints to achieve 94% picking success rate. Jiang et al. [10] requires 60k robot demos to learn its visual pick & place tasks. In contrast, our proposed method generates the distribution over the entire action space and shows its zero-shot learning ability on novel categories.

Few-shot manipulation requires the robot to manipulate in-distribution objects with a few demonstrations because robot demo collection is expensive. Many works focus on improving sample efficiency to enable few-shot policy learning. Transporter [20] exploits a rigid transformation prior in planar manipulation tasks. Works using equivariant models [21, 22, 23, 24, 25, 26, 27, 28, 29, 30] leverage symmetries in robotic tasks and have demonstrated its superior effectiveness for unseen pose generalization that further allows few-shot learning. However, these methods often only learn a single-task policy each time, which limits them from learning from a diverse dataset and generalizing to novel objects and tasks. In this paper, our model is able to learn a language-conditioned multi-task policy yet maintains high sample efficiency by leveraging the inherent symmetry in the language-conditioned manipulation problem.

Open-vocabulary manipulation represents a specific area in language-conditioned manipulation where the robot needs to generalize to out-of-distribution objects in a zero-shot manner. To perform diverse tasks in the open world, robots need to equip robust generalization ability because the majority of objects that robots encounter during deployment are unseen with novel poses, shapes, colors, or textures. Learning-from-scratch methods [20, 31, 32, 33, 34, 35, 36] perform well on seen objects but cannot generalize well on unseen ones. Using pre-trained models for robotic manipulation [37, 38] has shown the potential of giving robots commonsense knowledge distilled from internet-scale data. One popular approach is to combine VLMs with pre-trained skill functions. Rashid et al. [39] combines LeRF [40] with GraspNet [41] that allows zero-shot language-conditioned grasping. [42, 4] use LLM/VLMs as a zero-shot object detector and a text-level task planner. These methods usually assume access to a pre-trained library with robust skills. However, the lack of learning ability limits these methods to adapt to more complex task-orientated behaviors. For example, “insert the letter E block into the letter E hole” since there is no general actor available for placing skills. [43] uses NeRF-based dense semantic radiance fields and learns an NDF-style actor [26] on top of it to achieve high sample-efficiency. However, it requires task-relevant descriptors to define and regress the gripper pose, and cannot achieve zero-shot learning on unseen categories. Please refer to Table A6 for a detailed comparison. In this paper, we propose a novel approach that is capable of learning effective pick & place policies with a small amount of demonstrations while leveraging the zero-shot generalization ability from pre-trained VLM models.

3 Method

Problem Statement and Assumptions: This paper focuses on learning from demonstration for the planar language-conditioned pick-and-place. Given a set of demonstrations that contains observation-language-action tuples (o_t, ℓ_t, a_t), the objective is to learn a policy $p(a_t|o_t, \ell_t)$ where the action $a_t = (a_t^{\text{pick}}, a_t^{\text{place}})$ has pick and place components. The visual observation o_t is a top-down orthographic RGB-D reconstruction of the scene from several camera views. The pick and place components of action, a_t^{pick} and a_t^{place}, are parameterized in terms of $\mathrm{SE}(2)$ coordinates $(u, v, \theta_{\text{pick}})$ and $(u, v, \theta_{\text{place}})$, respectively, where u, v denotes the pixel coordinates of the gripper position, θ_{pick} is the pick orientation defined with respect to the world frame, and θ_{place} is the delta angle between the pick and place. The language instruction ℓ_t specifies the current-step instruction, e.g., “pick the red block and place onto the green and blue blocks” or “grasp the scissors by its handle and place into the brown box.” We assume ℓ_t for each step can be parsed into the pick instruction and the place instruction, $\ell_t = (\ell_t^{\text{pick}}, \ell_t^{\text{place}})$. For parser details, please see Appendix A.1.6.

Method Overview: There are three main modules. **(1)** The semantic module takes multi-view images $\mathcal{O}_t$ and the language instruction ℓ_t and outputs a dense semantic map that summarizes the visual and language input. **(2)** The language-conditioned pick module takes as input the raw RGB-D image of the scene o_t with the language instruction ℓ_t and outputs an action map over pick actions. **(3)** Similarly, the language-conditioned place module produces an action map over place actions. The only difference is the place module does the convolution with a crop-conditioned instead of a language-conditioned kernel.

3.1 Patch-level Semantic Module

The semantic module uses a pre-trained CLIP model to identify parts of the visual observation most relevant to the current task. Specifically, it takes the language goals ℓ_t^{pick} and ℓ_t^{place} and the current

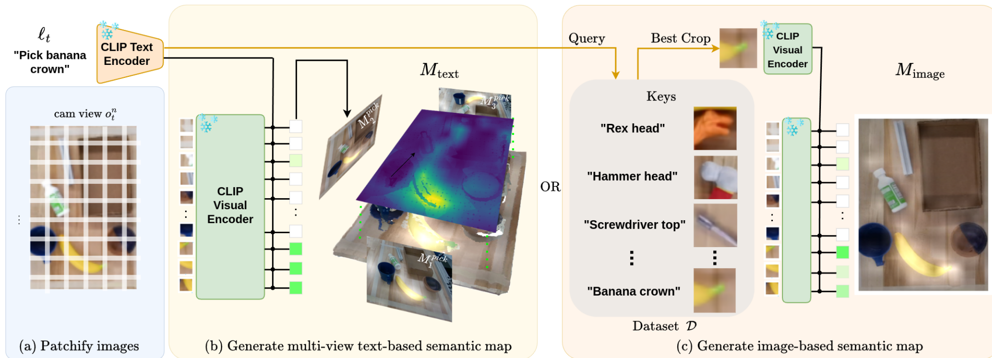


Figure 2: **Patch-level Semantic Map Extraction.** After (a) **patchifying images** into patches, two types of semantic maps are presented in this work. (b) **Text-conditioned maps** M_{text} allow open-vocabulary zero-shot generalization, which is a projection from the semantic point cloud constructed with multi-view semantic maps $\{M_{\text{text}}^i\}_{i=1}^N$. (c) **Image-conditioned maps** M_{image} enhances few-shot performance of the model.

N-view observations $\mathcal{O}_t = \{o_t^1, o_t^2, ..., o_t^N\}$ as input and produces semantic maps M_t^{pick} and M_t^{place} that highlight the language goals in the pixel space. Note that while these semantic maps do not tell the system exactly where to pick, they provide a strong visual-language prior to the pick and place modules. The semantic modules are illustrated in Figure 2 and are described in the following.

Text-conditioned Semantic Maps for Zero-shot Learning: We use CLIP [3], which was trained by minimizing the cosine similarity between the image feature and its text label with internet data, to generate a pixel-wise semantic score for each of the N views in $\mathcal{O}_t$. We split each image along a grid into image patches with patch size p and stride s. Each RGB image patch is then scored with its cosine similarity to the language instructions with pre-trained CLIP features. The text-conditioned semantic generation function $\mathcal{M}_{text}$ can be described by

$$\mathcal{M}_{\text{text}}(\mathcal{P}(o_t^n), \ell_t) = \mathcal{P}^{-1}(\mathcal{E}_t^{\text{patch}} \cdot \mathcal{E}_{\ell_t}^T), \tag{1}$$

where $\mathcal{P}$ denotes the image patchification function and $\mathcal{P}^{-1}$ denotes an inverse process that transforms all similarity scores back to the original image dimension. $\mathcal{E}_t^{patch} \in \mathbb{R}^{(m \times n) \times d_m}$ is the embedding outputs from the CLIP image encoder, where $(m \times n)$ and d_m denote the number of image patches and the output embedding dimension of CLIP respectively. $\mathcal{E}_{\ell_t} \in \mathbb{R}^{1 \times d_m}$ is the embedding output for language instruction ℓ_t from the CLIP text encoder. After getting pixel-wise semantic features for each of the N views, we integrate this information into a single point cloud and label each point with the corresponding semantic maps from all views so that we get a final top-down text-conditioned semantic map M_{text} via projection, as shown in panel (b) of Figure 2.

Image-conditioned Semantic Map for Enhancing Few-shot Learning: Although the text-conditioned map highlights image regions related to language instructions, we found it insufficient because the high-value region does not necessarily highlight the correct object if certain categories are underrepresented during CLIP training. This misalignment creates noisy training samples as shown in Figure A1. To solve it, we further introduce the image-conditioned semantic map, which is illustrated in Figure 2(c). Starting with the demonstrations, we identify the image crops in the demonstration data corresponding to pick and place events, where the event timing is determined by checking the gripper status. For all pick/place events identified, we store into a database a pair comprised of the image patch at the pick/place location and the language query that describes the pick/place object (left side of Figure 2(c)). Then, at inference time, we index into the dataset using the language query text and recall the corresponding image crop, e.g., recall the image crop from the dataset corresponding to "banana crown". The crop query process can be expressed by

$$\text{QueriedCrop}(\mathcal{D}, \ell_t) = \underset{\text{crop} \in \mathcal{D}}{\arg\max}(QK_{\text{crop}}^T), \tag{2}$$

where $K_{\text{crop}} \in \mathbb{R}^{N \times d_m}$ denotes all language embeddings that correspond with N image patches for all N pick and place objects in dataset $\mathcal{D}$. $Q \in \mathbb{R}^{1 \times d_m}$ denotes the embedding of the language query. Then, we generate image-conditioned semantic maps by evaluating the cosine similarity between the

CLIP embeddings of the recalled image crop and the patch embeddings from the top-down image o_t. The image-conditioned semantic generation function $\mathcal{M}_{\text{image}}$ can be described by

$$\mathcal{M}_{\text{image}}(\mathcal{P}(o_t), \ell_t, \mathcal{D}) = \mathcal{P}^{-1}(\mathcal{E}_t^{\text{patch}} \cdot \mathcal{E}_{\text{crop}}^T), \tag{3}$$

where $\mathcal{E}_{\text{crop}} \in \mathbb{R}^{1 \times d_m}$ denotes the embedding output for the image crop from the CLIP image encoder. If the pick/place target cannot be located in the dataset, i.e., $\max(QK_{\text{crop}}^T)$ is below a threshold, then the image-conditioned map function returns None. See Figure A1 for more image-conditioned semantic map examples and Appendix A.7 for implementation details.

Fuse Text/image-conditioned Semantic Maps: We first fuse the text/image-conditioned semantic map by a pixel-wise weighted averaging. Finally, we concatenate the top-down semantic map calculated above with a feature map produced by a convolutional encoder on the top-down depth image. This concatenated feature map gives downstream parts of the model precision information about object boundaries and shapes. The concatenated map is passed through a pixel-wise linear layer f_θ that produces a final semantic map output, M_{pick} or M_{place}. Given multi-view observations $\mathcal{O}_t$, language instruction ℓ_t, and dataset $\mathcal{D}$, the overall semantic function $\mathcal{M}$ can be expressed by

$$\mathcal{M}(\mathcal{O}_t, \ell_t, \mathcal{D}) = f_\theta(\frac{w_1 \mathcal{M}_{\text{text}}(\cdot) + w_2 \mathcal{M}_{\text{image}}(\cdot)}{w_1 + w_2}, \text{depth}), \tag{4}$$

3.2 Language-Conditioned Pick & Place:

The picking model f^{pick} calculates a probability distribution over gripper pose that corresponds to the probability of a successful grasp on the desired object part. This distribution $p(a_t^{\text{pick}} | o_t, \ell_t^{\text{pick}})$ is estimated by $f^{\text{pick}}(o_t, \ell_t^{\text{pick}}, M_t^{\text{pick}})$. The pick command actually executed by the robot is selected by $a_{\text{pick}}^\star = \arg\max a_t^{\text{pick}}$.

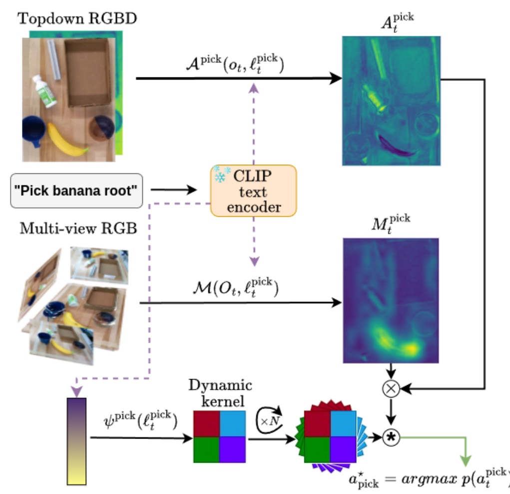


Figure 3: **Pick Module.** Our picking module consists of three branches. The top branch is our vision-language encoder $\mathcal{A}^{\text{pick}}$. The middle part is the semantic extractor $\mathcal{M}(O_t, \ell_t)$ that takes multi-view RGB observations with pick instruction and outputs picking semantic map M_t^{pick}. The bottom branch is the language-conditioned kernel generator, and we rotate the dynamic kernel to realize local $\mathrm{SE}(2)$ equivariance.

Symmetry of Language Conditioned Pick: The desired pick action is equivariant with respect to the pose of the object to be picked, i.e., $g \cdot p(a_t^{\text{pick}} | b^{\text{pick}}, \ell_t^{\text{pick}}) = p(a_t^{\text{pick}} | g \cdot b^{\text{pick}}, \ell_t^{\text{pick}})$, where b^{pick} denotes the object to be picked and $g \cdot$ denotes the action of a transformation g. Note that this form of equivariance is local to the object, in contrast to standard models that are equivariance with respect to the scene.

Specifically, assume the observation o_t contains a set of m objects $\{b_i\}_{i=1}^m$ on the workspace and denote the object b^{pick} as the goal object instructed by the language instruction ℓ_t^{pick}. If there is a transformation $g \in \mathrm{SE}(2)$ on the target object b^{pick} regardless of transformations on other objects, we denote it as $g \cdot o_t^{b^{\text{pick}}}$. The symmetry underlying f can be stated as

$$\arg\max f^{\text{pick}}(g \cdot o_t^{b^{\text{pick}}}, \ell_t^{\text{pick}}) = g \cdot \arg\max f^{\text{pick}}(o_t, \ell_t^{\text{pick}}) \tag{5}$$

Equation 5 claims that if there is transformation $g \in \mathrm{SE}(2)$ on the object b^ℓ, the best action $a_{\text{pick}}^\star$ to grasp the instructed object should be transformed to $g \cdot a_{\text{pick}}^\star$. If the symmetry is encoded in our pick model, it can generalize the pick knowledge learned from the demonstration to many unseen configurations. In the following, we use this symmetry to improve sample efficiency and generalization of our pick model. Please refer to Appendix A.9 for detailed proofs.

Pick Model Architecture: There are two main parts of the pick model. The first (shown in the top part of Figure 3) calculates a language-conditioned pick map as follows. We feed the raw RGB-D observation into a UNet, denoted as $\mathcal{A}^{\text{pick}}$, and encode ℓ_t^{pick} with the CLIP. The encoded language vector is concatenated onto the descriptor of every pixel in the bottleneck layer of $\mathcal{A}^{\text{pick}}$. The output of $\mathcal{A}^{\text{pick}}$ is denoted as $A^{\text{pick}}(o_t, \ell_t^{\text{pick}})$ or A_t^{pick} for simplicity. It is then integrated with the pick semantic map M_t^{pick} with element-wise multiplication, shown as $\otimes$ in Figure 3.

The second part of the pick model is the language-conditioned dynamic kernel, which is a *key novelty* in our approach. We leverage the language-conditioned symmetry by performing a cross-correlation between a language-conditioned dynamic kernel Φ and the feature map calculated above, as shown in Figure 3. The dynamic kernel Φ maps language embeddings to convolutional kernels that satisfy the steerability constraints [44]. It allows picking action inference to be $\mathrm{SO}(2)$ equivariant with respect to the object poses. Please refer to Appendix A.1.2 for our implementation details, Appendix A.8 for proof, and Appendix A.1.3 for dynamic kernel visualization.

Symmetry in Language-Conditioned Place: Place action that transforms pick target to the placement are bi-equivariant [45, 46, 47], i.e., independent transformations of the placement with g_1 and the pick target with g_2 result in a change ($a'_{\text{place}} = g_1 a_{\text{place}} g_2^{-1}$) to complete the rearrangement at the new configuration. Leveraging the bi-equivariant symmetries can generalize the learned place knowledge to different configurations and thus improve the sample efficiency [45, 46, 47]. The coupled symmetries also exist in the language-conditioned place:

$$\arg\max f^{\text{place}}(g_1 \cdot o_t^{b^{\text{place}}} + g_2 \cdot o_t^{b^{\text{pick}}}, \ell_t^{\text{place}}) = g_1\theta(g_2^{-1}) \cdot \arg\max f^{\text{place}}(o_t, \ell_t^{\text{place}}) \quad (6)$$

where $g_1 \cdot o_t^{b^{\text{place}}} + g_2 \cdot o_t^{b^{\text{pick}}}$ denotes $g_1 \in \mathrm{SE}(2)$ and $g_2 \in \mathrm{SE}(2)$ acting on the instructed placement b^{place} and the picked object b^{pick}, respectively. $\theta(g_2^{-1})$ denote the angle of the place action is rotated by $-g_2$. Specifically, the RHS of Equation 6 indicates that the best place location is rotated by g_1, and the place orientation is rotated by $\theta(g_1)\theta(g_2^{-1})$.[1] Our place model is designed to satisfy the language-conditioned equivariance of Equation 6. Detailed proofs can be found in Appendix A.9.

Place Model Architecture: Our language-conditioned place module is similar to the pick module. The place action distribution map is calculated as the cross-correlation between the semantic map M_t^{place} and the place dynamic kernel. The place dynamic kernel is generated with an image crop centered on the pick action as described above instead of the language embeddings. Implementation details can be found in Appendix A.1.4.

4 Experiments

4.1 Simulation Experiments

Tasks & Baselines For simulation tasks, we use 18 tasks provided by CLIPort Benchmark [8] for our simulation experiments. For baselines, we compare our method with three strong baselines: Transporter [20], CLIPort [8], VIMA [10]. Detailed descriptions of tasks and baselines can be found in Appendix A.10 and Appendix A.1.5.

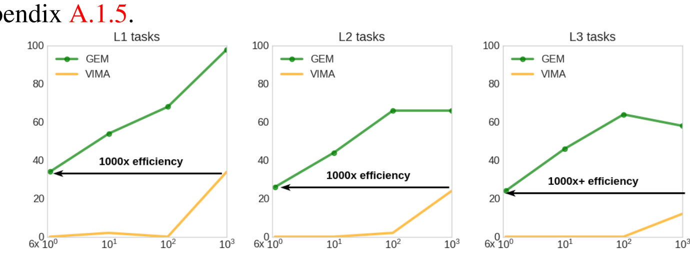


Figure 4: **Performance Comparisons on VIMABench [10].** X-axis and y-axis represent the number of demonstrations for training and task success rate during evaluation in the *visual_manipulation* task.

Simulation Results: In Table 1, we report the performance of our model and the baselines trained with $\{10, 20, 100\}$ demonstrations from CLIPort Benchmark [8]. We use "-multi" to denote the multi-task policy. The best performance is highlighted in bold in each column. Several conclusions can be drawn from Table 1. **(1)** GEM outperforms the baselines in all the tasks by a significantly large margin. For example, in task *separating-piles-unseen-colors*, our method gets 97.6% success rate with

[1]Please note the orientation component of a_{place} is the relative rotation between the pick and place pose.

Model	packing-box-pairs seen-colors			packing-box-pairs unseen-colors			packing-seen-google objects-seq			packing-unseen-google objects-seq			packing-seen-google objects-group			packing-unseen-google objects-group		
	10	20	100	10	20	100	10	20	100	10	20	100	10	20	100	10	20	100
Transporter-Lan [20]	50.4	72.4	86.8	41.2	40.4	60.7	36.3	57.0	83.2	31.7	46.8	54.9	49.6	57.0	81.2	56.6	59.4	77.4
CLIPort [8]	63.2	78.3	88.2	28.8	64.2	71.4	37.9	52.6	80.1	45.9	41.7	49.6	62.0	62.1	77.1	49.5	50.3	60.0
CLIPort-**multi** [8]	60.3	82.9	81.4	42.4	53.7	54.3	76.6	84.3	77.0	50.4	58.7	47.6	79.0	88.0	88.6	79.9	85.6	73.8
GEM (ours)	79.6	86.7	91.8	67.3	71.4	**78.2**	76.2	85.8	89.7	69.2	**79.8**	**86.0**	86.6	85.1	**94.2**	78.1	71.9	82.3
GEM-**multi** (ours)	**90.6**	**90.7**	**93.8**	**73.8**	**78.2**	**78.2**	**93.7**	**91.0**	**90.3**	**86.3**	79.7	75.7	**94.5**	**93.1**	**94.2**	**89.7**	**90.9**	**88.5**

Model	stack-block-pyramid seq-seen-colors			stack-block-pyramid seq-unseen-colors			separating-piles seen-colors			separating-piles unseen-colors			towers-of-hanoi seq-seen-colors			towers-of-hanoi seq-unseen-colors		
	10	20	100	10	20	100	10	20	100	10	20	100	10	20	100	10	20	100
Transporter-Lan [20]	52.0	72.7	94.3	18.0	26.0	17.0	40.0	60.0	92.0	56.0	73.8	52.3	81.1	88.6	95.7	43.4	48.3	60.0
CLIPort [8]	22.8	39.5	50.5	21.8	19.2	27.7	53.1	56.0	74.8	56.4	66.0	72.5	75.1	75.0	91.1	57.6	47.3	**99.4**
CLIPort-**multi** [8]	74.7	87.7	93.3	45.7	28.3	33.0	59.7	72.2	75.0	67.8	65.2	58.8	78.3	95.4	97.4	60.3	69.4	69.7
GEM (ours)	70.7	82.7	**96.3**	59.3	73.7	**84.3**	82.3	75.4	78.8	60.0	91.8	96.6	88.3	93.4	**100**	83.1	87.7	98.0
GEM-**multi** (ours)	**94.3**	**95.3**	95.0	**76.0**	**89.3**	78.7	**94.2**	**96.2**	**92.0**	**89.0**	**97.6**	**96.6**	**96.3**	**99.4**	98.9	**93.4**	**98.0**	97.1

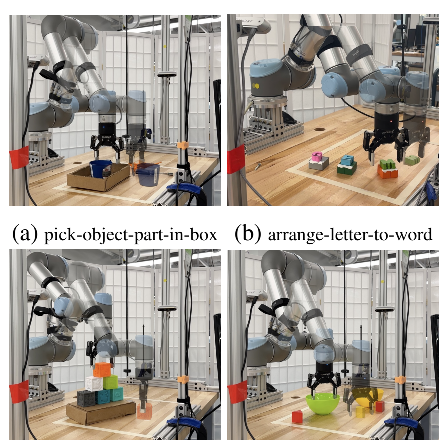
Model	align-rope			packing-unseen-shapes			assembling-kits-seq seen-colors			assembling-kits-seq unseen-colors			put-blocks-in-bowls seen-colors			put-blocks-in-bowls unseen-colors		
	10	20	100	10	20	100	10	20	100	10	20	100	10	20	100	10	20	100
Transporter-Lan [20]	11.5	33.7	72.4	24.0	26.0	30.0	26.4	39.2	58.4	20.0	24.8	23.6	42.7	68.7	86.3	12.0	17.0	36.0
CLIPort [8]	30.0	16.9	51.5	29.0	24.0	34.0	17.8	24.8	39.4	16.6	20.6	36.6	37.2	55.6	92.7	50.8	41.7	51.8
CLIPort-**multi** [8]	39.7	42.4	40.8	52.0	46.0	52.0	28.8	42.8	32.0	28.4	27.2	18.8	84.0	96.0	98.0	38.7	48.0	44.0
GEM (ours)	31.6	38.6	**69.0**	54.0	44.0	**52.0**	42.8	47.2	**62.4**	34.4	40.0	**62.8**	94.0	98.3	**100**	87.7	92.0	94.3
GEM-**multi** (ours)	**62.6**	**59.6**	58.6	**60.0**	**50.0**	**52.0**	**55.6**	**62.0**	56.8	**53.2**	**58.0**	46.4	**100**	**100**	**100**	**95.3**	**97.0**	**97.0**

Table 1: **Performance Comparisons on CLIPort Benchmark Tasks (%)** on 50 testing episodes. {10, 20, 100} denotes the number of demonstrations used in training. "**-multi**" denotes multi-task models where they are trained on 10 tasks and evaluated on each task separately. Best performances are highlighted in bold.

20 demos while the best baseline can only achieve 66.0%. **(2)** GEM is more sample efficient compared with the baseline. Trained with 10 demos, it can outperform the baselines with 20 and 100 demos on 10 out of 18 tasks. For instance, in *stack-block-pyramid-seq-seen-colors*, our method trained with 10 demos gets 70.7% success rate while CLIPort only gets 50.5% success rate trained with 100 demos. **(3)** GEM demonstrates strong zero-shot learning ability. The performance gap between GEM and the baselines becomes larger when tested with unseen colors and shapes. In *put-blocks-in-bowls* with 100 demos, the performance difference between our method and the CLIPort increases from $\Delta 7.3\%$ to $\Delta 42.5\%$ when tested on unseen colors. **(4)** GEM is capable of scaling up to learning a generalizable multi-task policy from a diverse dataset. As shown in multi-task results, GEM-multi performs best on 41 out of 54 evaluation cases. Overall, the results in Table 1 demonstrate the state-of-the-art sample efficiency and generalization ability of our proposed method. In Figure 4, we compare GEM with VIMA on VIMABench [10], where ours achieves the same performance with 6 demos comparing VIMA with 6k demos.

4.2 Real-world Experiments

For real-world experiments, we evaluate the few-shot and zero-shot learning ability of our model on two physical robot platforms: a table-top UR5 and a mobile Spot platform.



(a) pick-object-part-in-box (b) arrange-letter-to-word

(c) stack-block-pyramid (d) put-shapes-in-bowl

Figure 5: **Tabletop Tasks.**

Table-top Results: We evaluate our method in four tasks as shown in Figure 5 and report the results in Table 2. Objects and task descriptions can be found in Appendix A.11. Our single-task method outperforms the baseline on all tasks up to a margin of 87.5%. On *pick-object-part-in-brown-box*, our method trained with 5 demos reaches 92.5% success rate on seen objects while the baseline can only obtain 37.5% success rate. Besides, it shows strong generalization ability on unseen colors and shapes whereas the baseline fails to generalize well. For example, on the *arrange-letter-to-word* task, GEM can hit 75.0% success rate in arranging unseen

	pick-object-part-in-box			arrange-letter-to-word			stack-block-pyramid			put-shapes-in-bowl		
Model	1	3	5	1	3	5	5	15	20	5	15	20
CLIPort (seen)	37.5	47.5	37.5	5.6	50.0	50.0	33.3	43.3	76.7	0	25.0	62.5
CLIPort-multi (seen)	40.0	52.5	30.0	30.5	55.6	41.6	**50.0**	63.3	83.3	62.5	**87.5**	75.0
GEM (seen)	**80.0**	**75.0**	**92.5**	44.4	**66.7**	72.2	40.0	**80.0**	**93.3**	**87.5**	62.5	**87.5**
GEM-multi (seen)	47.5	67.5	82.5	**47.2**	44.4	**80.5**	**50.0**	60.0	83.3	50.0	75.0	75.0
CLIPort (unseen)	14.2	10.7	28.5	18.8	43.8	43.8	5.0	0	15.0	0	12.5	12.5
CLIPort-multi (unseen)	17.9	32.1	35.7	18.8	37.5	43.8	3.3	10.0	6.7	12.5	50.0	12.5
GEM (unseen)	**32.1**	35.7	**82.1**	**62.5**	56.3	**75.0**	30.0	**53.3**	63.3	12.5	**37.5**	**62.5**
GEM-multi (unseen)	17.9	**53.6**	**82.1**	25.0	**68.8**	68.8	**36.7**	40.0	**66.7**	**50.0**	**62.5**	**62.5**

Table 2: **Performance Comparisons on Real-world Tabletop Tasks (%)**. {1, 3, 5}, {5, 15, 20} are the numbers of demonstration episodes used in training. "(seen)" denotes that the model is evaluated on seen objects which are included in the training set and "unseen" means that the model is firstly trained on training objects and then is evaluated on novel objects. "-multi" denotes the multi-task model where one model is trained using all data across task. Best performances are highlighted in bold.

letters while the baseline can only achieve 43.8%. The experiments on the real robot further prove the few-shot and zero-shot learning ability of GEM. The main failure mode we find is that CLIP is extremely sensitive to colors compared with shapes, which results in the wrong semantic map. The detailed analysis can be found in Appendix A.4. We also found that the image-based semantic map is crucial for real-world performance because it helps reduce noise on text-based semantic maps. Please refer to Appendix A.3 for a detailed ablation. For the multi-task models, our model also outperforms the baseline on 20 out of 24 evaluations.

Mobile Manipulation Results: We also evaluate GEM on a Spot robot for language-conditioned pick and place tasks. We introduce an action parameterization trick (see Figure A12) so that the policy can generalize to a multi-table environment though all demos are collected on one table. For seen objects, our model reaches a success rate of 80%. For unseen objects, our model gets 50% success rate. A performance drop can be observed in our mobile manipulation results compared with tabletop experiments. The major reason is that the relative pose estimation between Spot and the table is inaccurate, which introduces a discrepancy when executing pixel-based actions.

(a) Tabletop (b) Mobile Manipulation

Figure 6: **Real-world Tabletop and Manipulation Setup.** Multi-view cameras are highlighted by red circles and workspaces are labelled by blue.

5 Conclusion

In this work, we analyze the inherent symmetry in language-conditioned manipulation and propose **G**rounded **E**quivariant **M**anipulation (**GEM**) that leverages such symmetry while preserving the zero-shot open-vocabulary ability via a novel technique to extract patch-level semantic maps from pre-trained VLMs. Our method is able to learn generalizable open-vocabulary manipulation policy from a limited number of demonstrations and achieves a high success rate on seen and novel objects. We demonstrate its few-shot and zero-shot ability with various simulated and real-world experiments. A limitation of our approach is that our action space is in $\mathrm{SE}(2)$ which limits its ability to perform more complex tasks. In the future, we will extend our method in $\mathrm{SE}(3)$ action space and enable a larger workspace with full mobility with on-robot cameras. Worth mentioning, the language-conditioned pick and place symmetries we studied in Section 3 and Appendix A.9 are also applicable for $\mathrm{SE}(3)$ action space, which provides a solid foundation to extend our method to $\mathrm{SE}(3)$ language-conditioned manipulation in the future using 3D convolution methods like [48, 46]. For ablation studies, equivariant proof, and implementation details, please see the following appendix.

References

[1] Y. Du, Z. Liu, J. Li, and W. X. Zhao. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936*, 2022.

[2] OpenAI. Gpt-4 technical report, 2023.

[3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[4] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023.

[5] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning*, pages 540–562. PMLR, 2023.

[6] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and A. Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

[7] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[8] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[10] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.

[11] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021.

[12] G. Tziafas, Y. Xu, A. Goel, M. Kasaei, Z. Li, and H. Kasaei. Language-guided robot grasping: Clip-based referring grasp synthesis in clutter. *arXiv preprint arXiv:2311.05779*, 2023.

[13] C. Tang, D. Huang, L. Meng, W. Liu, and H. Zhang. Task-oriented grasp prediction with visual-language inputs. *arXiv preprint arXiv:2302.14355*, 2023.

[14] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[15] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. *arXiv preprint arXiv:2306.14896*, 2023.

[16] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[17] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[19] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

[20] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

[21] D. Wang, R. Walters, X. Zhu, and R. Platt. Equivariant q learning in spatial action spaces. In *Conference on Robot Learning*, pages 1713–1723. PMLR, 2022.

[22] D. Wang, R. Walters, and R. Platt. SO (2)-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022.

[23] H. Huang, D. Wang, R. Walters, and R. Platt. Equivariant transporter network. *arXiv preprint arXiv:2202.09400*, 2022.

[24] H. Huang, D. Wang, X. Zhu, R. Walters, and R. Platt. Edge grasp network: A graph-based se (3)-invariant approach to grasp detection. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3882–3888. IEEE, 2023.

[25] L. Zhao, H. Li, T. Padir, H. Jiang, and L. L. S. Wong. E(2)-equivariant graph planning for navigation. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[26] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[27] J. Yang, C. Deng, J. Wu, R. Antonova, L. Guibas, and J. Bohg. Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation. *arXiv preprint arXiv:2310.16050*, 2023.

[28] L. Zhao, L. Kong, R. Walters, and L. L. Wong. Toward compositional generalization in object-oriented world modeling. In *ICML*, 2022.

[29] L. Zhao, X. Zhu, L. Kong, R. Walters, and L. L. Wong. Integrating symmetry into differentiable planning with steerable convolutions. In *ICLR*, 2023.

[30] J. Y. Park, O. Biza, L. Zhao, J. W. van de Meent, and R. Walters. Learning symmetric embeddings for equivariant world models. In *ICML*, 2022.

[31] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019.

[32] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[33] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[34] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[35] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. *6th Annual Conference on Robot Learning (CoRL)*, 2022.

[36] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. In *7th Annual Conference on Robot Learning*, 2023. URL `https://openreview.net/forum?id=hRZ1YjDZmTo`.

[37] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[38] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[39] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg. Language embedded radiance fields for zero-shot task-oriented grasping. In *7th Annual Conference on Robot Learning*, 2023. URL `https://openreview.net/forum?id=k-Fg8JDQmc`.

[40] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

[41] H.-S. Fang, C. Wang, M. Gou, and C. Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020.

[42] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[43] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola. Distilled feature fields enable few-shot language-guided manipulation. *arXiv preprint arXiv:2308.07931*, 2023.

[44] G. Cesa, L. Lang, and M. Weiler. A program to build e (n)-equivariant steerable cnns. In *International Conference on Learning Representations*, 2021.

[45] H. Huang, D. Wang, R. Walters, and R. Platt. Equivariant Transporter Network. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi:10.15607/RSS.2022.XVIII.007.

[46] H. Huang, O. Howell, X. Zhu, D. Wang, R. Walters, and R. Platt. Fourier transporter: Bi-equivariant robotic manipulation in 3d. *arXiv preprint arXiv:2401.12046*, 2024.

[47] H. Ryu, H.-i. Lee, J.-H. Lee, and J. Choi. Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning. *arXiv preprint arXiv:2206.08321*, 2022.

[48] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022.

[49] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[50] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[51] A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[52] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT Press, 2016.

[53] T. S. Cohen and M. Welling. Steerable cnns. In *International Conference on Learning Representations*, 2017.

[54] M. Weiler and G. Cesa. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019.

[55] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, et al. Simple open-vocabulary object detection. In *European Conference on Computer Vision*, pages 728–755. Springer, 2022.

[56] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia, C. Finn, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.

[57] T. S. Cohen and M. Welling. Steerable CNNs. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rJQKYt5ll`.

[58] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[59] O. Puny, M. Atzmon, H. Ben-Hamu, I. Misra, A. Grover, E. J. Smith, and Y. Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

[60] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.

[61] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.

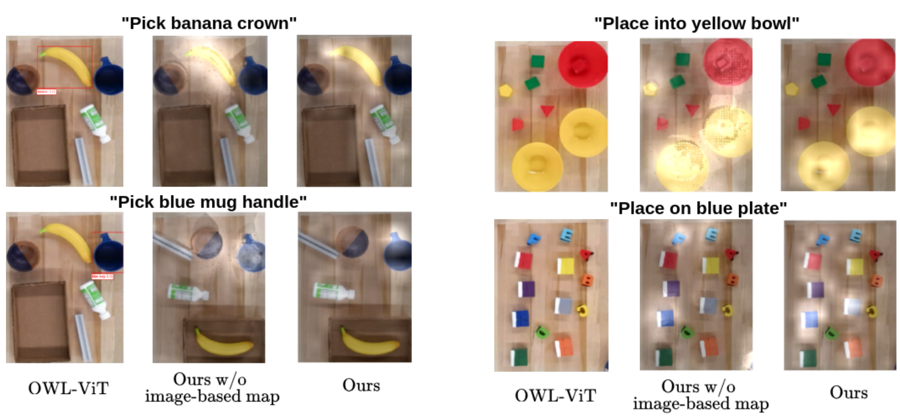
Appendix

In the appendix, we provide the following sections. Section A.1 introduces the implementation details for the semantic module, the picking module, and the placing module. Section A.2 and Section A.3 provide a detailed ablation study to show how every component in our method contributes to the performance. Section A.4 discusses the failure modes of our method. Section A.5 provides an analysis of how our method scales given more demonstrations. Section A.6 compares the task performance between two VLMs: CLIP [3] and GroundingDINO [49]. Section A.7 presents implementation details and text-text similarity experiments for the image-conditioned semantic map generation. Section A.8 gives a background of symmetry groups. Section A.9 introduces detailed proofs for the steerable language-conditioned kernels and its equivariance property. Section A.10 and Section A.11 provide task and object details for simulation and real-world experiments in the tabletop setting. Section A.12 includes implementation details for the mobile manipulation experiments. Section A.13 provides a discussion about how our method as a multi-task skill function can bridge high-level text-based planners and real-world manipulation policies.

A.1 Implementation Details

A.1.1 Patch-level Semantic Module

In the semantic module, we grid the image into image patches using patch size $p = 40$ and stride $p = 20$. For CLIP, we use OpenAI's *clip-vit-base-patch32* pre-trained model. To extract accurate semantics, we integrate semantic maps from multi-view camera views and do re-projection from the point cloud to get a top-down semantic map. We found three cameras work well both in simulation and in real-world experiments. Before fusing the semantic map and the action map, we first use a topdown depth image to refine the map which aims to provide objectness into the semantic map. The depth image is sent into a two-layer CNN encoder. Then, we concatenate the raw semantic map and the output from the shape encoder and the concatenated features are linearly projected into the final semantic map using a single CNN layer. We set the text-text similarity threshold to 0.965. For the weighted averaging of text/image-conditioned semantic maps, we set $w_1 = 0.8$ and $w_2 = 0.2$ for simulation experiments. For real-world experiments, we set $w_1 = 0.5$ and $w_2 = 0.5$.



(a) Pick Maps (b) Place Maps

Figure A1: **Pick&Place Semantic Extraction Comparison.** OWL-ViT often fails to find the object given open-vocabulary queries. Ours w/o image-conditioned semantic map is able to highlight image regions that correlate with language instructions but it is noisy. By fusing text-image-conditioned maps, the semantic maps perfectly align with the instructions.

We provide visualization in Figure A1. For picking, OWL-ViT fails to find the specific part of "banana crown". It can only find out the banana only when given "banana" as a whole. It also fails to find anything given "blue mug handle" until we change the prompt to "blue mug". On the other hand, our method can highlight correct objects. For placing, OWL-ViT fails to find either “yellow bowl”/“bowl”or “blue plate”. Our method can highlight correct objects.

A.1.2 Pick Module Implementation

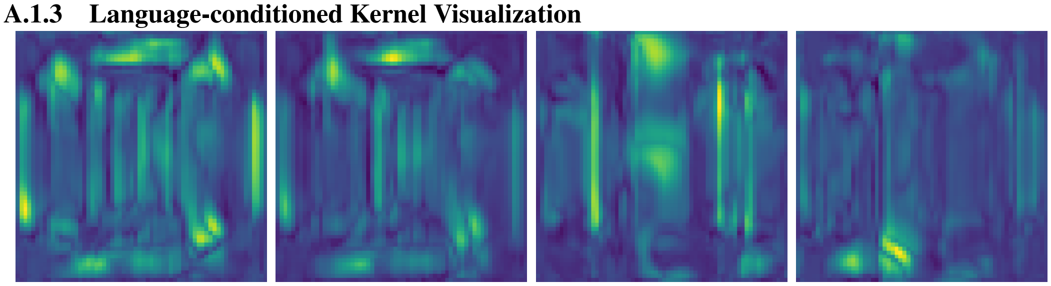
Model	#Parameters
GEM (ours)	5.7M
CLIPort	389M
VIMA	8M
Transporter-Lan	6.6M

Table A1: **Number of Trainable Parameters.** Our model is lightweight compared with baselines.

Our pick module is composed of two UNets [50]. Each UNet has 8 residual blocks and each block contains two convolution layers. The first four residual blocks trade spatial dimensions for channels with maxpooling in each block; the last four residual blocks upsample the feature embedding with bilinear-upsampling operations. ReLU [51] activations are interleaved inside the network. One UNet takes a 4-channel RGB-D image and the language feature. The other UNet takes the expanded language feature and outputs a three-channel square kernel $\mathbb{R}^{3\times h\times h}$. The kernel is rotated 180 times to $\mathbb{R}^{180\times 3\times h\times h}$, and we apply the Fourier Transform to the first dimension to get its Fourier representation $\mathbb{R}^{F\times 3\times h\times h}$. After the cross-correlation, 72 rotations are uniformly sampled with inverse Fourier Transform per pixel. The place module shares the same architecture as the pick module.

We feed the language embedding to a UNet ψ^{pick} and then rotate the output with a group of n rotations $\{\frac{2\pi i}{n}|0 \leq i < n\}$. This results in a stack of n rotated feature maps, $\Psi(\cdot) = \{g_0 \cdot \psi(\cdot), g_2 \cdot \psi(\cdot), \cdots, g_{n-1} \cdot \psi(\cdot)\}$, where $g_i = \frac{2\pi i}{n}$. Above each pixel, there is an n-dimension orbit-traversing signal. We apply the Fourier transform pixelwisely to the channels of $\Psi(\cdot)$ which preserves the channel distribution of each pixel of $\Psi(\cdot)$ as a set of Fourier coefficients that can approximate continuous $\mathrm{SO}(2)$ signals. In other words, the Fourier transform outputs a distinct vector of Fourier coefficients for each pixel in the feature map. The output is a dynamic steerable kernel $\Psi^{\text{pick}}(\ell_t^{\text{pick}})$ (we will write Ψ_t^{pick} for simplicity). The dynamic steerable kernel is cross-correlated with the dense feature map from the top to generate the pick action distribution, $p(a_t^{\text{pick}}) = (A_t^{\text{pick}} \otimes M_t^{\text{pick}}) * \Psi_t^{\text{pick}}$, where $\otimes$ denotes elementwise multiplication and $*$ denotes 2D convolution. Finally, an inverse FT is applied to return to the spatial domain. Notice that since Ψ_t^{pick} is represented with the Fourier coefficients, the cross-correlated result is also in the Fourier space. An arbitrary number of rotations can be sampled with inverse Fourier transform based on the task precision requirement. Notice that this model is equivariant with respect to rotations and translations of the object in $\mathrm{SO}(2) \ltimes \mathbb{R}^2$. $\mathbb{R}^2$ translational equivariance is achieved due to the property of cross-correlation [52]. The $\mathrm{SO}(2)$ rotation equivariance is achieved by the steerability [53] of the dynamic kernel $\Psi^{\text{pick}}(\ell_t^{\text{pick}})$.

A.1.3 Language-conditioned Kernel Visualization



(a) “blue block” (b) “red block” (c) “scissor blade” (d) “scissor handle”

Figure A2: **Visualization of Language-conditioned Adaptive Kernels.** Given different language instructions, our language-conditioned kernel generator generates language-conditioned adaptive kernels.

In our picking module, the picking kernel generator $\psi(\ell^{pick})$ is conditioned on language. A desired property of such kernels is that they are adaptive with respect to different language inputs.

We visualize four picking kernels of our multi-task model trained on data from four tabletop tasks in Figure A2. The kernels show interesting patterns where the kernels look similar given similar language instructions “blue block” and “red block”. And, given different language instructions like “scissor blade” or “scissor handle”, the kernels show different patterns.

A.1.4 Place module architecture

The language-conditioned place module is very similar to the pick module described earlier. Instead of A^{pick}, we have a distinct place model, A^{place}. Also, the model is conditioned on the place language ℓ_t^{place} rather than the pick language. Perhaps the biggest difference is that the dynamic kernel is now conditioned on the place image crop rather than the pick language. That is, instead of evaluating $\psi^{\text{pick}}(\ell_t^{\text{pick}})$, we evaluate $\psi^{\text{place}}(c_t)$ where c_t is an image crop centered on the position of the pick action calculated as described in the section above.

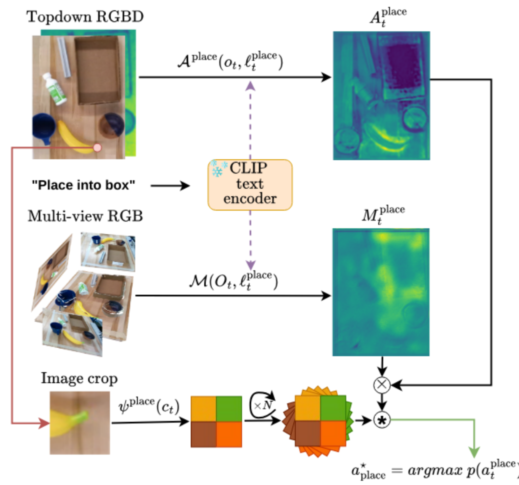


Figure A3: **Place Module.** The architecture of placing module is similar to picking module except that (1) the placing kernel generator $\psi^{\text{place}}(c_t)$ is conditioned on the image crop c_t centered at the previous picking action a_t, (2) $\psi^{\text{place}}(c_t)$ is constrained to be fully rotational-equivariant with E2CNN [54].

A.1.5 Baselines

For Transporter [20], it was originally a visual-only model. To encoder language information, we concatenate an additional language embedding obtained from the CLIP text encoder onto the bottleneck of its UNet-style affordance prediction module for both pick and place. We denote it as *Transporter-Lan*. CLIPort [8] uses the pre-trained CLIP model (both the vision encoder and the language encoder) with a trainable two-branch architecture. It fuses pre-trained language and visual features by pixel-wise multiplication and 1×1 convolution between different feature maps in its trainable layers. For fair comparison, all baselines use parsed language instructions and conduct data augmentations. The number of trainable parameters of each model is reported in Table A1.

For VIMA [10], we use the 8M model for training and evaluation. We train VIMA from sractch only on the *visual_manipulation* task in VIMABench [10] since *visual_manipulation* is the only task of which its prompt is in the *“pick something and place it into something”* form. Other tasks that include goal-image-conditioned settings are out of scope of this paper. Because VIMA does not use image data augmentation in its original paper, for fair comparison, we also do not perform data augmentation for the VIMA & GEM comparison in Figure 4. Moreover, we replace the depth channels in our method into segmentation images given that VIMA assumes access to ground truth object masks. For results in Figure 4, we train our method for 100 epochs and VIMA for 70 epochs. We only use image-conditioned semantic map in VIMABench for fair comparison because VIMA takes multimodal prompts where all object names are represented by object images.

A.1.6 Noun Parser

The instruction parser assumption can be easily removed with some high-level interpreters, e.g., LLMs. Our model will split this policy into $p(a_t^{\text{pick}}|o_t, \ell_t^{\text{pick}})$ and $p(a_t^{\text{place}}|o_t, \ell_t^{\text{place}}, a_t^{\text{pick}})$ and represent them as two different neural networks. We can reconstruct the full policy using the product rule, $p(a_t^{\text{pick}}, a_t^{\text{place}}|o_t, \ell_t^{\text{pick}}, \ell_t^{\text{place}}) = p(a_t^{\text{place}}|o_t, \ell_t^{\text{place}}, a_t^{\text{pick}})p(a_t^{\text{pick}}|o_t, \ell_t^{\text{pick}})$, where we assume a_t^{pick} is conditionally independent of ℓ_t^{place} given ℓ_t^{pick} and that a_t^{place} is conditionally independent of ℓ_t^{pick} given a_t^{pick}. Note that this policy can solve one-step tasks as well as multi-step tasks. In our experiments, the baselines and our methods have access to a ground truth parser that parses out object names from language instructions. We tested GPT-4 [2] to demonstrate the effectiveness of GPT-4 as a noun parser from novel natural language instruction. The parsing success rate is 100% for seen instructions and 99% for novel language instructions.

A.2 Ablation Study

To investigate the functionality of each component of our model, we present a detailed ablation study as shown in Table A2. We conduct the ablation study in two different tasks (stack-block-pyramid-seq-seen/unseen-colors, packing-seen/unseen-google-objects-group) with 10, 100 demonstrations. For each task, we train a single-task model and evaluate the mean reward at 30k SGD steps. We remove different design choices from GEM separately to analyze its importance: **(1)** GEM without semantic map: we remove the entire semantic extraction module for this variation. **(2)** GEM without multi-view extraction: we only use one top-down RGB image for semantic map extraction. **(3)** GEM without steerable kernels: we directly use the unrotated output from $\psi^{\text{pick}}(\ell^{\text{pick}})$ and $\psi^{\text{place}}(\ell^{\text{place}})$ to do 2D convolution. **(4)** GEM without image-based semantic map: we only rely on the text-based semantic map without using image-based semantic map in this case. **(5)** GEM without language parsing: we feed the entire language instruction to the model without parsing it to the ℓ^{pick} and ℓ^{place}. **(6)** GEM without language-conditioned attention module: we remove the language input for attention module $\mathcal{A}^{pick}$ and $\mathcal{A}^{place}$.

Table A2 summarizes the ablation experiment results. Below we discuss our findings: **(1)** Using semantic maps improves performances for all tasks and especially for unseen tasks. It indicates that the generalization ability of our model to novel objects mainly comes from our semantic map. **(2)** Multi-view semantic extraction is also vital for getting accurate semantic maps. Without multi-view extraction, *pyramid-unseen-100* drops 50%. **(3)** Without leveraging $\mathrm{SO}(2)$ symmetry provided by the steerable kernels, the model fails to complete all tasks because the model leverages no rotation equivariance. **(4)** The image-based semantic map partnering with the text-image semantic map can benefit policy learning. **(5)** Parsing the language instruction to ℓ^{pick} and ℓ^{place} slightly helps the policy learning for our model. **(6)** Using the language feature in the attention module $\mathcal{A}^{pick}$ and $\mathcal{A}^{place}$ introduces an inductive bias, especially for the unseen tests.

	pyramid-seen-100	packing-seen-100	pyramid-unseen-100	packing-unseen-100
Ours	**94.0**	89.1	**84.3**	86.2
w.o Semantic map	91.7 (↓ 2.3)	87.3 (↓ 1.8)	21.0 (↓ 63.3)	53.8 (↓ 32.4)
w.o multi-view	63.0 (↓ 31.0)	83.6 (↓ 5.5)	34.3 (↓ 50.0)	65.5 (↓ 20.7)
w.o Steerable kernel	7.7 (↓ 84.0)	64.0 (↓ 25.1)	1.7 (↓ 82.6)	46.0 (↓ 40.2)
w.o image-based map	91.3 (↓ 2.7)	83.9 (↓ 5.2)	82.7 (↓ 1.6)	**92.2** (↑ 6.0)
w.o Lan Parsing	93.0 (↓ 1.0)	86.7 (↓ 2.4)	84.3 (—)	90.2 (↑ 4.0)
w.o Lan-conditioning	90.7 (↓ 2.7)	**89.3** (↑ 0.2)	61.3 (↓ 23.0)	77.3 (↓ 8.9)

Table A2: **Ablation Study.** Arrows indicate the performance difference between ours and each other ablation variation. All variations are evaluated at 30k training steps with 50 testing episodes.

A.3 Ablation of Image-based Semantic Map in Real World

	arrange-letter-to-word	pick-object-part-in-box
our (seen)	**72.2**	**92.5**
w/o image map (seen)	42.2	92.5
our (unseen)	**75.0**	**82.1**
w/o image map (unseen)	68.8	50.0

Table A3: **Ablation on Image-based Semantic Map in Real-world Tasks.** For both tasks, 5 demos are used for training. We evaluate at 20k SGD steps. The image map denotes the image-based map.

We find that the image-based semantic map plays a crucial role in improving real-world performance as shown in Table A3. The hypothesis is that the text-based semantic map can be noisy for specific language instructions. Figure A1 shows that the text-based semantic fails to highlight the correct object given fine-grained instructions like "pick banana crown". In this case, the picking action has

	object-sorting (seen)	object-sorting (unseen)
GEM (ours)	12/15	6/12

Table A5: **Results on Real-world Mobile Manipulation Tasks for Seen and Unseen Objects.** Each of the 15 seen objects and 12 novel objects is tested with the pick & place instruction.

a misalignment with the high-value regions in the generated semantic map. These demonstrations serve as "bad" data points during training because these samples force the model to ignore the semantic guidance provided by the semantic map. If such demonstrations commonly exist in the dataset, our model will learn to ignore the guidance from semantic maps during the evaluation. By adding image-based semantic maps, we can ensure that the action points always align with high-value regions in the corresponding semantic map during training. Hence, our model will trust the semantic map and try to take actions on high-value regions with high semantic scores, which allows zero-shot generalization on novel objects highlighted by the semantic map during evaluation. As shown in Figure A1 and Figure A1, ours without image-based semantic map (middle), i.e. text-based semantic map is noisier than the one with image-based semantic map. For example, the banana crown is highlighted more accurately than the text-based map which only highlights the banana as a whole. For placing, it also helps better reduce the color sensitivity of CLIP as shown in the "yellow bowl" example in Figure A1, where the high value is suppressed after adding the patch semantic map. In the simulation ablation A2, image-based semantic maps do not have such a huge influence presumably because the text-based semantic maps are often accurate in simulation.

A.4 Failure Case Analysis

A.4.1 Tabletop Experiments

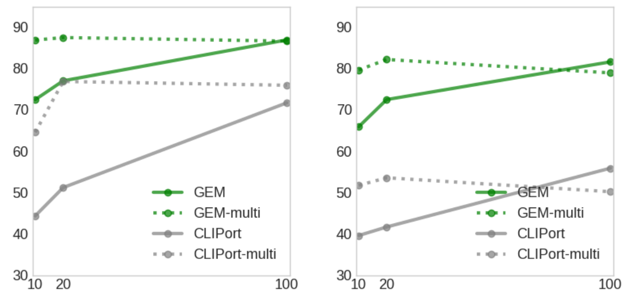
CLIP is highly sensitive to colors. Given an instruction like "pick up the yellow screwdriver", the CLIP map will be more likely to highlight all the yellow objects rather than all screwdrivers. Especially when there is a bright yellow block and a dark yellow screwdriver, the color sensitivity of CLIP biases the semantic map to give a higher value to the "yellower" objects and occasionally guides our model to pick up the bright yellow block. Adding more data is one way to alleviate this color bias because our vision-language encoder can learn to give more credit to shapes when yellow objects are all equally highlighted by CLIP. By using image-based semantic maps introduced in Section 3.1, it also reduces such color-sensitivity noise. For example in Figure A1, given instruction "blue plate", our method highlights plates in the scene while ours w/o image-based semantic map incorrectly highlights the blue letter F as well.

Task	#demo=1	#demo=3	#demo=5	#demo=10
arrange-letter-seen[1]	44.4	66.7	72.2	**83.3**
arrange-letter-unseen[1]	62.5	56.3	75.0	**81.5**

Task	#demo=5	#demo=15	#demo=20	/
block-in-bowl-unseen[2]	12.5 (**40.0**)	37.5 (**20.0**)	62.5 (**70.0**)	/
stack-pyramid-unseen[2]	30.0 (**40.0**)	53.3 (**60.0**)	63.3 (**86.67**)	/

Table A4: **Additional Results for Real-world Experiments.** Dataset size and lightning conditions could affect real-world performance. By adding more data[1] and fixing lightning issues[2], the performance of our method increases. **Bold numbers** denoted the updated results.

Dataset size and lightning conditions could affect real-world performance. By adding more data[1] and fixing lightning issues[2], the performance of our method increases. The results are in Figure A4.



(a) Zero-shot (seen) tasks (b) Few-shot (unseen) tasks

Figure A5: **Data Scalability in Simulation Tasks.** We visualize the average success rates across all simulation tasks as data increases. Our method has more capacity as well as a higher success rate compared with the baseline.

A.4.2 Spot experiments

As stated in Section 4.2, most failure cases come from calibration errors when transforming pixel actions into real 3D actions. And, we observe the same color sensitivity of the CLIP-based semantic map where it often tends to highlight colors rather than shapes given an instruction like *"pick the yellow screwdriver"*.

A.5 Scalability

For real-world tasks, we collect more robot data in *arrange-letter-to-word* to demonstrate the data scalability. As shown in Figure A4, with more data, the model performance keeps increasing.

For real-world industrial applications, a key question is the scalability of our method because it will get access to more data.

In this section, it shows that our method scales with more data. And, it is capable of multi-task learning and has the potential to benefit from a bigger multi-task dataset. Data scalability in simulation is shown in Figure A5. Given more data, the single-task model keeps getting better performance across all tasks. Meanwhile, multi-task models perform better than single-task in the low-data region. The result shows the data scalability and multi-tasking scalability of our method. An interesting finding is that the zero-shot tasks are converging to a lower overall success rate compared with few-shot tasks for two reasons: (1) zero-shot performance are constrained by the open-vocabulary ability of CLIP which sets a hard performance upper bound; (2) few-shot (seen) tasks are scalable given that it is considered to be "close-vocabulary" because all objects appeared at least once in the training set.

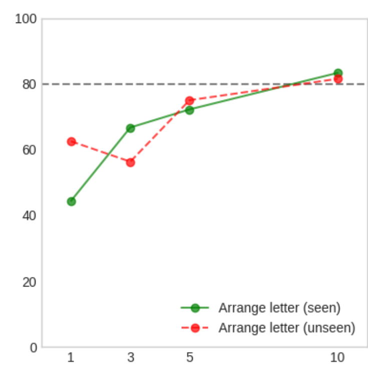
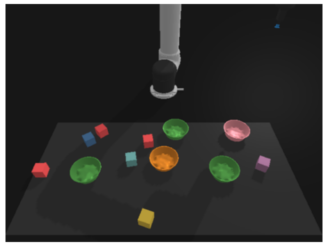


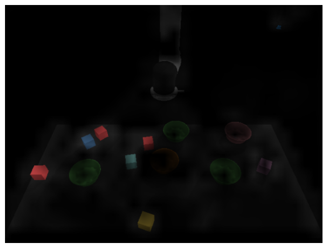
Figure A4: **Data scalability in arrange-letter-to-word in real-world.** The x-axis denotes the number of demonstrations. The y-axis denotes the success rate. Given 10 demonstrations, the success rate increases for both few-shot (seen) and zero-shot (unseen) settings.

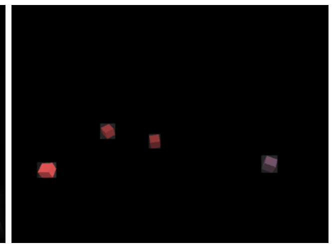
A.6 Semantic Extraction from Different VLMs

We compare a popular open-vocabulary object detector OWL-ViT [55] which is used to ground language for robotic tasks in [56]. For OWL-ViT, we use *owlvit-base-patch32* and set score_threshold



(a) RGB observation



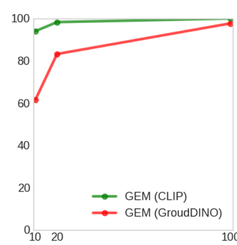


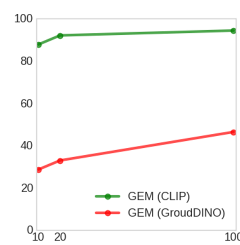
(b) CLIP (ours) (c) Grounding DINO

Figure A6: **Semantic map visualization using different VLMs for "red blocks".** CLIP (ours) creates a more uniform semantic map than Grounding DINO which is strongly biased by the objectness.

to 0.1. OWL-ViT fails to detect any object given "banana crown", "blue mug handle", "yellow bowl", and "blue plate" as shown in Figure A1 and Figure A1. OWL-ViT is able to find out banana given the prompt "banana" instead of "banana crown". Blue mug can be detected given "blue mug" but nothing detected given "blue mug handle". However, it fails to detect the yellow bowls whether using "yellow bowl" or "bowl".

We also compare our method with another recent zero-shot open-vocabulary detection method. i.e. Grounding DINO [49]. In Figure A6, Grounding DINO shows a stronger objectness bias where our method using CLIP generates a more uniform semantic map. In Figure A7, we compare GEM (CLIP) and GEM (Grounding DINO) with patch-level maps generated by CLIP (ours) and object-level maps generated by Grounding DINO. The results show that Grounding DINO reaches similar performance compared with CLIP in seen tasks. However, its performance drops dramatically in unseen tasks. Our hypothesis is that Grounding DINO has a strong "objectness" inductive bias. If the Region Proposal Network in Grounding DINO fails to propose correct regions that include the desired object, the performance drops. *Given the comparable performance in seen tasks, it shows that exploring more VLM variations is an interesting future direction.*





(a) block-in-bowl-seen (b) block-in-bowl-unseen

Figure A7: **Task Success Rate Using Different VLMs.**

A.7 Query Image Crops from Dataset via Text-text Similarity

We calculate the text-to-text cosine similarities using CLIP's text encoder between the given object name and all objects in the dataset to retrieve the corresponding image crop as introduced in Section 3.1. We set the text similarity threshold to be 0.965. If the returned text similarity is above the threshold, the corresponding image crop can be successfully retrieved. We found 0.965 is robust enough to exclude all incorrect objects in the dataset while adding certain free-form language adaptability. For example, as shown in Figure A8, given a language instruction "big bottle middle", our method can not only retrieve the correct image crop by identifying “big bottle middle”, but can also retrieve images labeled by synonyms like “big bottle body” if such a datapoint exists in the dataset.

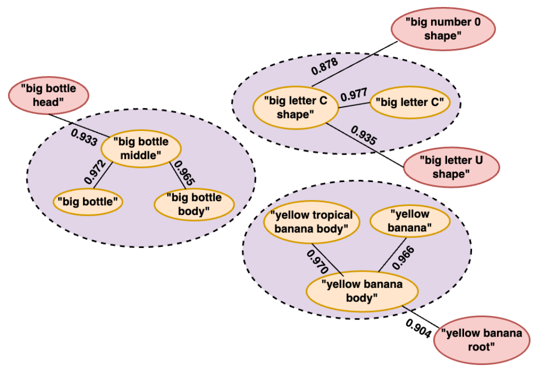


Figure A8: **Query from Dataset.** The yellow circles denote the objects and the purple circle denotes the words that are considered synonyms given a threshold of 0.965. The numbers on the connection lines show the text-text similarity scores. The word with scores bigger than the threshold is considered a successful query and vice versa.

A.8 Background on Symmetry Groups

A.8.1 Group and Representation

In this work, we are primarily interested in the $\mathrm{SO}(2)$ group and cyclic group C_n. $\mathrm{SO}(2)$ contains the continuous planar rotations $\{\mathrm{Rot}_\theta : 0 \le \theta < 2\pi\}$. $C_n = \{\mathrm{Rot}_\theta : \theta \in \{\frac{2\pi i}{n} | 0 \le i < n\}\}$ contains only rotations by angles which are multiples of $2\pi/n$. A d-dimensional *representation* $\rho\colon G \to \mathrm{GL}_d$ of a group G assigns to each element $g \in G$ an invertible $d{\times}d$-matrix $\rho(g)$. Different representations of $\mathrm{SO}(2)$ or C_n help to describe how different signals are transformed under rotations.

1. The trivial representation $\rho_0\colon \mathrm{SO}(2) \to \mathrm{GL}_1$ assigns $\rho_0(g) = 1$ for all $g \in G$, i.e. no transformation under rotation.
2. The standard representation
$$\rho_1(\mathrm{Rot}_\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$
represents each group element by its standard rotation matrix. Notice that ρ_0 and ρ_1 can be used to represent elements from either $\mathrm{SO}(2)$ or C_n.
3. The regular representation ρ_{reg} of C_n acts on a vector in $\mathbb{R}^n$ by cyclically permuting its coordinates $\rho_\lambda(\mathrm{Rot}_{2\pi/n})(x_0, x_1, ..., x_{n-2}, x_{n-1}) = (x_{n-1}, x_0, x_1, ..., x_{n-2})$.
4. The irreducible representation ρ^i_{irrep} could be considered as the basis function with the order/frequency of i, such that any representation ρ of G could be decomposed as a *direct sum* of them. Signals defined on the group $\mathrm{SO}(2)$ can be decomposed as limits of linear combinations of complex exponential functions $(\sin, \cos)$.

A.8.2 Feature Vector Field.

We formalize images and 2D feature maps as feature vector fields, i.e., functions $f\colon \mathbb{R}^2 \to \mathbb{R}^c$, which assign a feature vector $f(\mathbf{x}) \in \mathbb{R}^c$ to each position $\mathbf{x} \in \mathbb{R}^2$. The action of an element $g \in \mathrm{SO}(2)$ on f is a combination of a rotation in the domain of f via ρ_1 (this rotates the pixel positions) and a transformation in the channel space $\mathbb{R}^c$ (i.e., fiber space) by $\rho \in \{\rho_0, \rho_1, \rho_\lambda, \rho_{\mathrm{irrep}}\}$. If $\rho = \rho_0$, the channels do not change. If $\rho = \rho_{\mathrm{reg}}$, then the channels cyclically permute according to the rotation. If $\rho = \rho_{\mathrm{irrep}}$, then the channels shift.

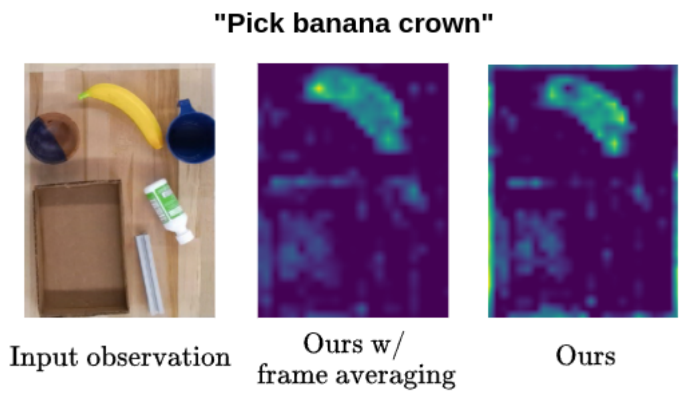


Figure A9: **Rotational equivariance of semantic map.** The patch size and stride is set to 20 and 10 respectively.

We denote this action (the action of g on f via ρ) by $T_g^\rho(f)$:

$$[T_g^\rho(f)](\mathbf{x}) = \rho(g) \cdot f(\rho_1(g)^{-1}\mathbf{x}). \tag{7}$$

A.8.2.1 Equivariant Mapping

A function $f : X \to Y$ is considered to be SE(2)-equivariant if it can commutes the action of the SE(2) group $f(T_g^x \cdot x) = T_g^y \cdot f(x)$ for all $g \in \mathrm{SE}(2)$, where T_g^x and T_g^y defines the group element g acts on the input and output of the function f. We sometimes omit the action space of g and denote it as $f(g \cdot x) = g \cdot f(x)$.

A.8.3 Steerable Kernel

The most equivariant mappings between spaces of feature fields are realized by convolutions with G-steerable kernels [57]. The G-steerable kernels are convolution kernels $K : \mathbb{R}^n \to \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ satisfying the *steerability constraint*, where n is the dimensionality of the space, d_{out} and d_{in} are the output and input field type

$$K(g \cdot x) = \rho_{\text{out}}(g) K(x) \rho_{\text{in}}(g)^{-1} \tag{8}$$

A.8.4 Language Steerable Kernel

Given a 2D square tensor κ with the size of $\mathbb{R}^{h \times h}$, rotating κ with a group of n rotations $\{\frac{2\pi i}{n} | 0 \leq i < n\}$ results a steerable kernel K with $\rho_{\text{in}} = I$. It was proved in [57, 58, 46]. The shortest answer is that a rotation g applied to κ (i.e., $g \cdot x$) on the LHS of Equation 8 is equivalent to a channel permutation (i.e., ρ_{out} on the RHS of Equation 8) of K with the unrotated κ.

A.8.5 Equivariant Property of Semantic Maps

To guarantee a strict local equivariance property of our output action, it requires not only the steerable kernels but also the attention maps and the semantic maps to be locally equivariant with respect to object movements in the input image. Firstly, UNet is known to be good at preserving geometric features from its input due to the residual connections mechanism. However, it is not clear whether the image-based semantic map also has such an equivariance-preserving property. We investigated this problem and found that CLIP is rotational invariance to a certain degree. Given a fixed language instruction ℓ and one image o_t, the similarity score $f_{CLIP}(o, \ell)$ is very close to the score $f_{CLIP}(g \cdot o, \ell)$, where $g \in \mathrm{SO}(2)$ rotates the image globally. Given this image-level rotational invariance property, it helps preserve the local equivariance when extracting patch-level semantic maps. To prove the point, we construct an equivariant version of our semantic map using the frame averaging technique introduced by [59] and compare it with a normal semantic map in Figure A9. By calculating the error between the equivariant map and a normal semantic map, we get an absolute mean error of 0.02 which indicates that our semantic map preserves the local equivariance to some degree. Worth mentioning, we also found the equivariance error increases when we increase the patch size for semantic extraction.

A.9 Theory and Proofs

A.9.1 Steerable kernel for realizing local equivariance

Our picking model is consist of language-conditioned kernel generator κ^{pick} and observation net ϕ^{pick} and can be written as

$$f^{\text{pick}}(o_t, \ell_t^{\text{pick}}) = \kappa(\ell_t^{\text{pick}}) * \phi(o_t, \ell_t^{\text{pick}}) \tag{9}$$

Picking symmetry is realized by language-conditioned kernel

$$\begin{aligned} \arg\max(g \cdot \kappa(\ell_t^{\text{pick}}) * \phi(o_t, \ell_t^{\text{pick}})) \\ = \\ g \cdot \arg\max(\kappa(\ell_t^{\text{pick}}) * \phi(o_t, \ell_t^{\text{pick}})) \end{aligned} \tag{10}$$

The placing module is implemented as follows

$$f^{\text{place}}(o_t, \ell_t^{\text{place}}, c_t) = \kappa^{crop}(c_t) * \phi(o_t, \ell_t^{\text{place}}) \tag{11}$$

And placing symmetry is realized by crop-conditioned kernel

$$f^{\text{place}}(o_t, \ell_t^{\text{place}}, c_t) = \kappa^{crop}(c_t) * \phi(o_t, \ell_t^{\text{place}}) \tag{12}$$

A.9.2 Equivariance proof for language steerable kernel

Proposition 1 *if $\kappa(\ell_t)$ is a steerable kernel, it approximately satisfies the symmetry stated in Equation 5.*

Intuitively, if ϕ is an identity mapping, the cross-correlation between a steerable kernel and the o_t captures the exact symmetry. That is any transformed b^l will be cross-correlated at one pixel location with the steerable kernel. Detailed proof of Proposition 1 can be found in the following section. **Translational Equivariance.** Since FCNs are translationally equivariant by their nature, if the target object b^ℓ is translated to a new location, the cross-correlation between $\kappa(\ell_t) * \phi(o_t, \ell_t)$ will capture this translation and there is no change in the change space.

Rotation Equivariance. Assuming ϕ satisfies the equivariant property that $\phi(T_g^0 o_t, \ell_t) = T_g^0 \phi(o_t, \ell_t)$ and the rotation of b^ℓ is represented by $T_g^0 o_t$, we start the proof with lemma 1 and lemma 2.

Lemma 1 *if $k(x)$ is a steerable kernel that takes trivial-type input signal, it satisfies $T_g^0 K(x) = \rho_{\text{out}}(g^{-1})K(x)$.*

Prove Lemma 1. $\rho_0(g)$ is an identity mapping. Substituting ρ_{in} with $\rho_0(g)$ and g^{-1} with g in Equation 8

$$\begin{aligned} T_g^0 K(x) &= K(g^{-1}x) \\ &= \rho_{\text{out}}(g^{-1})K(x)\rho_{\text{in}}(g) \\ &= \rho_{\text{out}}(g^{-1})K(x) \end{aligned}$$

Lemma 2 *Cross-correlation satisfies that*

$$(T_g^0(K \star f))(\vec{v}) = ((T_g^0 K) \star (T_g^0 f))(\vec{v}) \tag{13}$$

Prove Lemma 2. We evaluate the left-hand side of Equation:

$$T_g^0(K \star f)(\vec{v}) = \sum_{\vec{w} \in \mathbb{Z}^2} f(g^{-1}\vec{v} + \vec{w})K(\vec{w}).$$

Re-indexing the sum with $\vec{y} = g\vec{w}$,

$$= \sum_{\vec{y} \in \mathbb{Z}^2} f(g^{-1}\vec{v} + g^{-1}\vec{y})K(g^{-1}\vec{y})$$

is by definition

$$\begin{aligned} &= \sum_{\vec{y} \in \mathbb{Z}^2} (T_g^0 f)(\vec{v} + \vec{y})(T_g^0 K)(\vec{y}) \\ &= ((T_g^0 K) \star (T_g^0 f))(\vec{v}) \end{aligned}$$

as desired.

Given Lemma 1 and lemma 2, we can prove that

$$\begin{aligned} \kappa(\ell_t) * \phi(T_g^0 o_t, \ell_t) =& \kappa(\ell_t) * T_g^0 \phi(o_t, \ell_t) \\ =& \kappa(\ell_t) * T_g^0 \phi(o_t, \ell_t) \\ =& T_g^0 T_{g^{-1}}^0 \kappa(\ell_t) * T_g^0 \phi(o_t, \ell_t) \\ =& T_g^0 [T_{g^{-1}}^0 \kappa(\ell_t) * \phi(o_t, \ell_t)] \text{ lemma 2} \\ =& T_g^0 [\rho_{\text{out}}(g)\kappa(\ell_t) * \phi(o_t, \ell_t)] \text{ lemma 1} \end{aligned}$$

It states that if there is a rotation on o_t, the grasp position is changed by T_g^0, and the rotation is changed by $\rho_{\text{out}}(g)$. Since the cross-correlation is calculated for each pixel without stride, the rotated b^ℓ is captured by $\rho(g)$. In our implementation, we generate the language-conditioned steerable kernel $\kappa(\ell_t)$ but remove the constraint of the equivariant property of ϕ. However, the U-Net architecture with the long skip connection can maintain the equivariance a little bit, and extensive data augmentation is used to force the model to learn the equivariance.

A.9.3 Proof of the Steerability of $\mathcal{L}(\psi(\cdot))$

$$\begin{aligned} \mathcal{L}(T_g^0 \psi(\cdot)) =& T_g^0 \{T_{g_1}^0 \psi(\cdot), T_{g_2}^0 \psi(\cdot) \cdots, T_{g_n}^0 \psi(\cdot)\} \quad g_i \in C_n \\ =& \{T_{gg_1}^0 \psi(\cdot), T_{gg_2}^0 \psi(\cdot) \cdots, T_{gg_n}^0 \psi(\cdot)\} \\ =& \{T_{g_2}^0 \psi(\cdot), T_{g_3}^0 \psi(\cdot) \cdots, T_{g_n}^0 \psi(\cdot), T_{g_1}^0 \psi(\cdot)\} \text{ if } g = g_1 \\ =& \rho_{\text{reg}}(g^{-1}) \mathcal{L}(\psi(\cdot)) \end{aligned}$$

Since $\mathcal{L}(T_g^0 \psi(\cdot)) = \mathcal{L}(g^{-1}x)$, we achieve that $\mathcal{L}(g^{-1}x) = \rho_{\text{reg}}(g^{-1})\mathcal{L}(x)$. Substituting g^{-1} with g shows that $\kappa(c) = L(\psi(\cdot))$ satisfies the steerability constraint shown in Equation 8 and it is a steerable kernel with regular-type output and trivial-type input. Since Fourier transformation on the channel space maps the discrete $\mathrm{SO}(2)$ signal above each pixel to the coefficients of the basis function. It realizes an irreducible steerable kernel that has trivial-type input and irrep-type output [54, 44].

A.10 Simulation Tasks

The simulator inherits the design of Ravens-10 [20]. It has 3 cameras (topdown, left, right) pointing towards a rectangular workspace. Each camera provides a 480x640 RGB-D image that can be used for a top-down RGB-D reconstruction. Each task owns an oracle agent that can generate the expert action given the current language instruction and the observation.

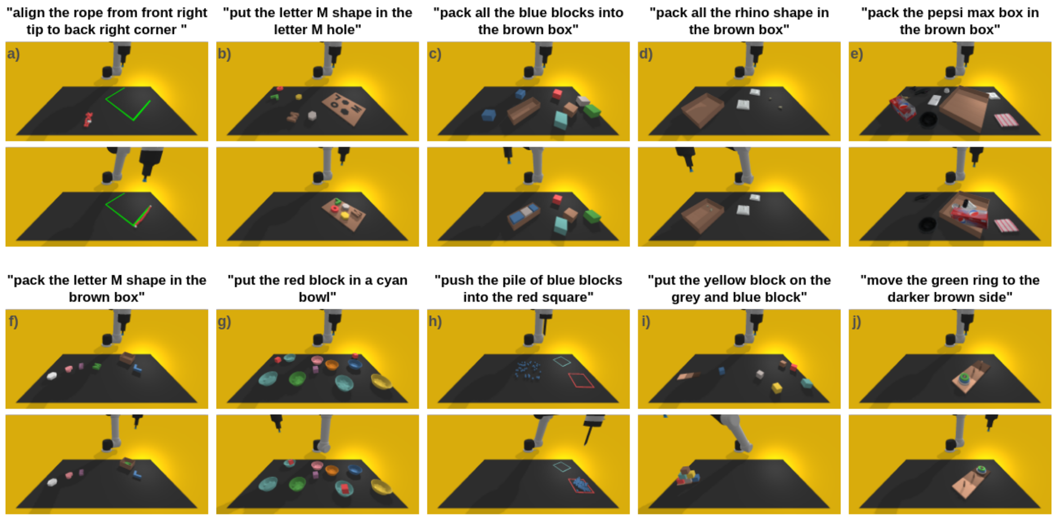


Figure A10: **CLIPort benchmark tasks.** We use tasks from the CLIPort benchmark [8] for evaluating our method in simulation. For each task, we provide one initial scene (upper image) and one final state (bottom image) with one specific language instruction example. In each scenario, one or more language instructions may be involved to finish the task. The tasks are defined as followed: (a) align-rope, (b) assembling-kits-seq-seen/unseen-colors, (c) packing-box-pairs-seen/unseen-colors, (d) packing-seen/unseen-google-objects-groups, (e) packing-seen/unseen-google-objects-seq, (f) packing-unseen-shapes, (g) put-blocks-in-bowls-seen/unseen-colors, (h) separating-tiles-seen/unseen-colors, (i) stack-block-pyramid-seq-seen/unseen-colors (j) towers-of-hanoi-seq-seen/unseen.

The simulation experiment contains 18 tasks from CLIPort benchmark [8]. Tasks that include the "google" identifier sample objects from a subset of the Google-scanned dataset [60] which contains 56 different objects. All 56 Google objects are separated into "seen-google" objects set with 37 objects and "unseen-google" objects set with 19 objects. For "seen-google" task variations, the training and testing objects are all sampled from the full google set. For "unseen-google" task variations, the training objects are from seen-google set, and testing objects are sampled from unseen-google set. For "seen/unseen-color" tasks, CLIPort benchmark defines a seen-color set that contains seven colors {red, green, blue, yellow, brown, gray, cyan} and an unseen-color set {red, green, blue, orange, purple, pink, white}. These two sets share three colors {red, green, blue}. For "seen-color" task variations, the colors of the training and testing object are all sampled from the seen-color set. For "unseen-color" task variations, the colors of training objects are from seen-color set, and testing objects are sampled from unseen-color set. Tasks (b) and (f) share a geometric object set which contains 20 objects like "letter A shape", "pentagon", "star". The geometric shape set is divided into a seen-shape set and an unseen-shape set that contains 14 and 7 objects respectively. Refer to [8] for more details. Specific task details are provided as follows.

(a) **Align-rope**: The instruction template is *"Align the rope from {direction}"*. The objective of this task is to connect two end-points of a rope between 2 corners of a 3-sided square.

(b) **Assembling-kits-seq-seen/unseen-colors**: The instruction template is *"Put the {color} {object} in the {location} {object} hole"*. This task requires the agent to pick an object and place it into a hole with the same shape. For example, "pick the green letter R shape and place into the green letter R block hole."

(c) **Packing-box-pairs-seen/unseen-colors**: The instruction template is *"Pack all the {colors} blocks into the brown box"*. The robot will be asked to pick blocks of two specific colors, the robot needs to identify all blocks with such colors and place them into the brown box. There are also blocks of other colors that serve as distractors.

(d) **Packing-google-objects-group-seen/unseen-colors**: The instruction template is *"Pack all the {object} in the brown box"*. For each step, the robot will be asked to pick a specific

Model	learning from demos	few-shot	zero-shot	GT low-level skills	GT objectness required	Pre-trained model
ViLA [4]	✗	✗	✗	teleoperation	not required	GPT-4V
VoxPoser [5]	✗	✗	✓	required*	OWL-ViT	GPT-4V, OWL-ViT, SAM
MOO [56]	✓	✗	✓	not required	OWL-ViT	OWL-ViT
VIMA [10]	✓	✗	✗	not required	Mask RCNN	T5, Mask RCNN
CLIPort [56]	✓	✓	✗	not required	not required	CLIP
GEM (ours)	✓	✓	✓	not required	not required	CLIP

Table A6: **Comparison Among Language-conditioned Manipulation Methods.** Our model allows few-shot and zero-shot generalization without ground truth training, object detectors, or segmentation models other than per-trained CLIP. We define "few-shot" as the learning ability to reach a reasonable task success rate given less than 20 demonstrations, and "zero-shot" as policy generalization on unseen objects. "GT objectness" means the method needs robust object detector or segmentation models during training or testing. "GT low-level skills" denotes whether the method assumes access to low-level policies that map pixels to actions. *Skill fine-tuning via demonstrations available.

object and place into the box. In the scene, there are at least two objects in this category and at least two distractors from other categories. The robot needs to pick and place all the objects as instructed in the scene to finish the task.

(e) **Packing-google-objects-seq-seen/unseen-colors**: The instruction template is *"Pack the {object} in the brown box"*. In this task, the agent is asked to pick the objects and place them in the brown box in a specific order based on the language descriptions. The robot needs to pick and place in the correct order as instructed.

(f) **Packing-unseen-shapes**: The instruction template is *"Pack the {object} in the brown box"*. Training objects are samples from the geometric shape set and the seen color set. During evaluation, objects are randomly sampled from the shape set, and the color is sampled from the unseen color set.

(g) **Put-blocks-in-bowl-seen/unseen-colors**: Instruction template is *"Put the {color} blocks in a {color} bowl"*. The agent is asked to pick the block with the instructed color and place it into the bowl. All the blocks are in the same shape.

(h) **Separating-piles-seen/unseen-colors**: The instruction template is *"Push the pile of {block color} blocks into the {square color} square"*. In this scenario, there are two square zones with different colors and a stack of blocks with one specific color. One of the zones is considered as a distractor. The task asks the agent to push the pile of blocks in certain colors into a specific zone.

(i) **Stack-block-pyramid-seen/unseen-colors**: The instruction template is *"Put the {pick color} block on {place color}"*. The robot needs to stack a 3-2-1 block pyramid by following step-by-step language instructions. At the beginning of each episode, six colored blocks are generated randomly and one plate with three colors is also placed in the workspace to indicate placing locations for the first three blocks.

(j) **Towers-of-hanoi-seq-seen/unseen-colors**: The instruction template is *"Move the {object} ring to the {location}"*. In this scenario, there is one peg base and three rings of different sizes. The peg base also contains three stands. The objective of the task is to train the robot to pick the specific ring and place it into the correct peg stand.

A.11 Real-world Table-top Tasks

Setting: As shown in Figure A12, we use a UR5 robot arm with Robotiq gripper for the table-top setting. There are one Microsoft Kinect Azure Camera and two Realsense D455 cameras mounted around a 29cm $\times$ 21cm workspace to capture the multi-view RGB-D images. The topdown RGB-D observation has a size of 320 $\times$ 240 pixels to cover the workspace. We select CLIPort as our real-world baseline since it performs the best among the baselines in simulated tasks.

(a) pick-object-part-in-box (b) arrange-letter-to-word (c) stack-block-pyramid (d) put-shapes-in-bowl

Figure A11: **Tabletop object set.** The transparent arm shows the picking action and the solid arm shows a successful placing action.

Tasks: We design 4 tasks with language instructions for our physical experiments to measure the performance of zero-shot learning and few-shot learning. Each task is tested with seen objects and unseen objects. Figure 5 shows the training and evaluation object set for different tasks. Our diverse object sets cover in-category objects, novel objects with unseen textures, unseen colors, and unseen shapes.

i) **pick-object-part-in-brown-box:** As shown in Figure 5, for each step, the robot is given a language instruction e.g., "pick the blue mug handle and place into brown box" and it needs to pick the specific part of objects instructed by the language instruction and place it into a brown box. In this task, there are 10 objects and each object has 2 parts, e.g., "mug brim" and "mug handle" are two parts for the object "mug". The instruction template is *"Pick the {object} and place into brown box"*. In this task, the agent is asked to pick objects and place them into a box based on language instructions. The object is not only counted for picking as a whole but two specific parts on each object are expected to be picked, which increases the complexity of the task. For the unseen part, the open-world object sets are used for evaluation.

ii) **arranging-letter-to-word:** A step-by-step instruction is given to the model like "pick blue letter E block and place onto green plate". The instruction template is *"Pick the {color} letter {letter} and place on {color} plate"*. This task aims to test the text recognition capability of our model. The agent was trained to pick up differently colored letter blocks and place them on colored plates. To improve orientation adaptability, black and white lines are painted on all alphabet blocks and plates to indicate the correct orientation of certain letters. A success rate of 0.5 was counted if the letter was placed on the correct plate but with a wrong orientation. Unseen letters and numbers are also employed in the evaluation to test the model's zero-shot ability.

iii) **block-stacking-pyramid:** The robot needs to stack a 3-2-1 block pyramid using color blocks. For each step, the instruction is similar to "pick yellow block and place on gray and red block." To complete the task, the robot needs to successfully finish the pyramid following the instructions. The instruction template is *"Pick the {pick color} block on {place color1} and {place color2}) block"*. If the robot is stacking the first pyramid layer, a plate with three different colors is placed in the workspace to indicate placing locations. For these steps, the instructions template is *"Pick the {pick color} block on {place color1} plate"*. In this task, the primary goal is to construct a pyramid using 6 blocks. The process involves stacking 6 colored blocks into a 3-2-1 pyramid in each episode. Three of these blocks are chosen to form the base of the pyramid, and three colored, planar squares are used to determine the placement position and orientation. Instructions for placing the other three blocks on top are given in the format, "Pick color A block and place on color B and C blocks." And the unseen version of this task where the evaluation involves blocks that have not been seen before.

iv) **pick-shapes-in-bowl:** As shown in Figure 5, for each episode, given an instruction like "pick the yellow pentagon block and place into green bowl", the robot needs to rearrange the pentagon block into the green bowl. The instruction template is *"Pick the {color} {shape} and place into {color} bowl"*. The goal of this task is to test the model's ability to recognize different colors and shapes. The agent is instructed to select a block that matches a specific color and shape and place it into a bowl with color. The model is tested on both seen and new colors and shapes.

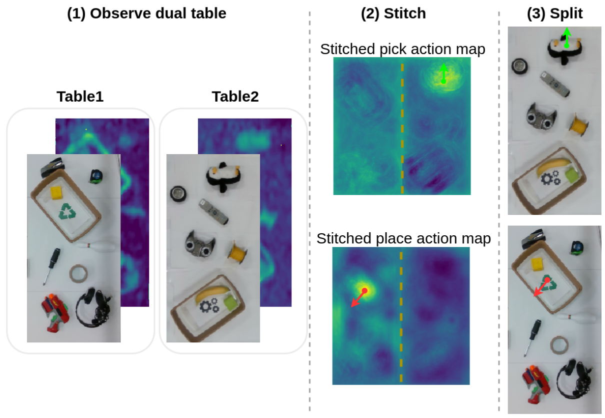


Figure A12: **The Stitch-and-split Trick in Mobile Manipulation.** The first column shows observations on two separate tables. By stitching the output action maps and then taking argmax for actions (step 2), we enable our method to directly generalize to multiple workspaces. Please note the instructed pick target and the place target are located in two different tables for this observation.

Training and Evaluation Details: For each task, we collect a data set of n expert demonstrations, where each demonstration contains a sequence of one or more $(o_t, \ell_t, \bar{a}_t^{\text{pick}}, \bar{a}_t^{\text{place}})$. $\bar{a}_t^{\text{pick}}$ and $\bar{a}_t^{\text{place}}$ denotes the expert pick action and place action. We use them to generate one-hot pixel maps as the ground truth labels for the pick module and the place module. The model is trained with cross-entropy loss end to end and we train our pick model and the place model separately. For both our method and baselines, we train each model for a total number of 30k SGD steps and evaluate the performance every 10k steps. Apart from training a single-task policy per method, we also train a multi-task policy for our methods and CLIPort [8]. Numbers of demonstrations for multi-task training are defined to be that we separately sample (10, 20, 100) from each task. For example, GEM**-multi** with 10 demonstrations is one model trained with a dataset that contains a total of 100 demonstrations sampled from all ten tasks with their seen object sets and color sets. We train the multi-task models for 300k SGD steps and evaluate every 100k steps. We report the best performance in these three evaluations per model for each task.

We measured the performance in the same way as used in CLIPort [8]. The metric is in the range of 0 (failure) to 100 (success). Partial rewards are calculated in multi-step tasks. For instance, in the task of pushing colored piles into the colored square, pushing 10 piles out of 50 into the correct zone will be credited $\frac{10}{50} \times 100\%$ rewards.

In our semantic module, we set the $patch\_size$ and $stride$ as 40 and 20 to generate the semantic map for each side-view image. We combine the text-based and image-based semantic maps with a weighted sum (0.2:0.8).

Training and Evaluation: Demonstrations are manually collected by humans and each demonstration is defined as a one-time completion of the task. For instance, in *pick-object-part-into-brown-box*, one demo contains 20 pick&place actions where each object part is demonstrated once. For *block-stacking-pyramid-seq*, one demo includes six pick&place actions to finish one 3-2-1 block pyramid.

We train a single-task policy and a multi-task policy for our model and the baseline with different numbers of demonstrations. Single-task models and multi-task models are trained for 20k and 100k SGD steps respectively. The performance is measured with seen and unseen objects separately. For each test, we randomly place seen and unseen objects in the workspace and the configurations are different from those in the training set. We run 20 evaluations per task per model.

Figure A13: **Real-world Mobile Manipulation Object Set.** Left is the training object set and right is the unseen test set.

A.12 Mobile Manipulation

For open-world manipulation, mobility is a must because the robot needs to move in an unstructured world. We evaluate our model on a mobile manipulation platform to demonstrate an interesting generalization case. With the translational equivariance of CNN, we can deploy our model directly to an arbitrary number of workspaces even if the data is only collected in one workspace. As shown in Figure A12, our model takes the images of two workspaces as inputs, and we can use the same pick kernel and place kernel to do the cross-correlation with the dense feature map of each workspace concurrently. The action can be queried with spatial argmax across two tables.

The instruction in the *object-sorting* task is "pick {object_name} and place into {symbol_name} box". We collected 5 pick and place demonstrations for each object in our training object set. With 15 training objects, there is a total of 75 pick-and-place actions. There are two boxes for placing objects: a "gear box" and a "recycle box". During the evaluation, there are 12 unseen objects and we replace the "gear box" with a "smile face box" as a novel box during evaluation.

Setting: We use a Boston Dynamics Spot robot with an arm for the mobile manipulation setting. There are two 106 cm $\times$ 53 cm tables in the environment. For calibration simplicity, we use three Realsense D435 cameras for each table to get multi-view images of the workspace. The topdown RGB-D observation has a size of 320 $\times$ 160 pixels to cover each table. Each table is attached with an Apriltag [61] and the Spot could commute between two tables by detecting its relative pose to the tag. We leave the full mobility implementation without the AprilTag for future work.

Task: We design an object-sorting task where the robot needs to do the pick & place between two tables. We do not designate the pick table and the place table. Objects and boxes are randomly placed on two tables. Given language instructions like "pick black headphone and place into recycle box", the robot needs to pick up the correct object from one table and place it into the correct box. As shown in Figure A13, our training object set contains 15 objects and two boxes with a recycle symbol and a gear symbol. For the unseen object set, we have 10 novel objects and one novel box with a happy face symbol.

Training and Evaluation: We collect 5 demos for each object and train our model for 50k SGD steps. During the evaluation, we evaluate two scenarios: **(1)** seen objects with novel spatial positions and orientations and **(2)** unseen objects with random positions in the workspace. Given a language instruction, it is considered a success only if the robot picks and places the correct object into the correct box as instructed. During evaluation, we randomly initialize objects and boxes in the workspace. We evaluate pick and place for each object in our object set.

A.13 Bridging Text-based Planner and Real-world Manipulation via Language-condition Policy:

Acknowledging the impressive reasoning ability of LLMs, a language-conditioned manipulation policy can serve as a bridge between a high-level reasoning machine and a physical agent. In this

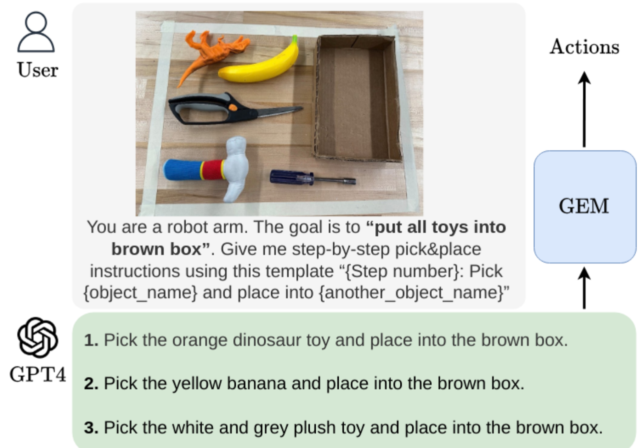


Figure A14: **Our Language-conditioned Policy Bridges LLM-level Planning and Real-world Manipulation.** GPT-4 takes observation and a vague language goal and breaks it into step-by-step specific instructions that can be executed successfully by our model in the real world.

section, we test our model with LLMs to solve semantically complicated and long-horizon tasks. As shown in Figure A14, we design a vague language goal, i.e., “pick all toys and place into brown box” and ask LLM to understand the goal and break it into step-by-step pick-and-place instructions. Our method then takes the step-by-step instruction to execute the action in the real world. Figure A14 shows a real example of how can our method take advantage of LLMs like GPT-4 [2] to directly enable long-horizon policies in real-world tasks. We test our multi-task model with “pick all toys and place into brown box” in the real world. With GPT-4’s instructions, our model can pick up all three toys in three steps. Without GPT-4, it only picks up the toy hammer and fails to pick up other toys.