ABM: Attention before Manipulation

Fan Zhuo[1,2] , **Ying He**[2] , **Fei Yu**[1,2*] , **Pengteng Li**[2] ,
Zheyi Zhao[1,2] and **Xilong Sun**[1]
[1]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)
[2]College of Computer Science and Software Engineering, Shenzhen University, China
zhuofan2022@email.szu.edu.cn, heying@szu.edu.cn, yufei@gml.ac.cn,
lipengteng2021@email.szu.edu.cn, zhaozheyi@gml.ac.cn, sunxilong1988@163.com

Abstract

Vision-language models (VLMs) show promising generalization and zero-shot capabilities, offering a potential solution to the impracticality and cost of enabling robots to comprehend diverse human instructions and scene semantics in the real world. Existing approaches most directly integrate the semantic representations from pre-trained VLMs with policy learning. However, these methods are limited to the labeled data learned, resulting in poor generalization ability to unseen instructions and objects. To address the above limitation, we propose a simple method called "Attention before Manipulation" (**ABM**), which fully leverages the object knowledge encoded in CLIP to extract information about the target object in the image. It constructs an Object Mask Field, serving as a better representation of the target object for the model to separate visual grounding from action prediction and acquire specific manipulation skills effectively. We train ABM for 8 RLBench tasks and 2 real-world tasks via behavior cloning. Extensive experiments show that our method significantly outperforms the baselines in the zero-shot and compositional generalization experiment settings.

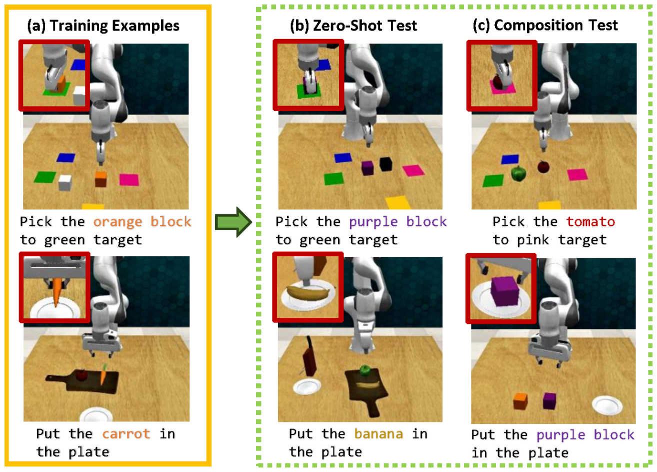
(a) Training Examples
(b) Zero-Shot Test
(c) Composition Test
Pick the orange block to green target
Pick the purple block to green target
Pick the tomato to pink target
Put the carrot in the plate
Put the banana in the plate
Put the purple block in the plate

Figure 1: The illustration of our motivation. (a) shows that we train our model and baselines on a subset of objects, instructions and actions. (b) shows that we evaluate models in zero-shot experiment setting where objects and instructions are unseen in (a). (c) shows that we evaluate models in compositional experiment setting where the target objects originate from different tasks in (a).

1 Introduction

The primary goal of robot learning is to perform various tasks in the workspace [Paradis *et al.*, 2021; Zhang *et al.*, 2018; Rahmatizadeh *et al.*, 2018]. To meet the requirements of human-robot interaction, robots need to comprehend the relationship between human instructions and the environment, and then execute accurate actions to complete tasks. Recent studies propose to use instructions and visual images as inputs to optimize end-to-end models through imitation learning objectives [Stepputtis *et al.*, 2020]. However, such methods introduce a challenge where end-to-end training tightly couples instructions, perceptions, and actions, *i.e.*, models only learn the mapping between semantic features and robot actions. To be specific, these methods perform well when perceptions and instructions share the same distribution as the training data (see Figure 1 (a)). However, they fail to generalize to unseen objects and instructions, which create a novel distribution different from the training data (see Figure 1 (b)). Furthermore, they also struggle to complete compositional tasks (see Figure 1 (c)), whose target objects originate from different training tasks.

Many studies demonstrate the powerful generalization and zero-shot abilities of vision-language models (VLMs) which are pre-trained at the internet scale. Recent works [Shridhar *et al.*, 2022; Liu *et al.*, 2022] treat the encoders of VLMs as feature extractors to train a model with certain generalization capabilities and have shown remarkable achievements in robot generalization tasks. Though success, directly integrating semantic representations from VLMs into policy optimization still fails to effectively decouple the binding relationships among instructions, perceptions, and actions. These approaches result in the model achieving only feature-level generalization. In other words, the model performs well only when presented with instructions and images semantically close to the training data. Otherwise, the model almost fails.

[*]Corresponding Author.

The reason is that the semantic representation distribution of the unseen images and instructions is not consistent with that of the training data, which is an unknown distribution to the model and thus cannot be predicted correctly. Consequently, these methods can achieve only limited generalization, failing to meet the requirements of open-ended human instructions and object sets. Therefore, we need to explore a more intuitive and effective representation to leverage the rich commonsense within VLMs, aiding the policy in achieving better generalization performance.

To address this issue, we propose "Attention before Manipulation" (**ABM**), which effectively decouples visual grounding from action prediction by extracting rich commonsense within CLIP, enabling generalization to unseen objects and instructions. Different from previous works, we design an Object Mask Field, which indicates the spatial location of the target object in the point cloud of current scene, serving as a better representation of the target object to assist in policy learning. Given a human instruction, we first break down the instruction into a target object description and a robot skill via instruction decomposition module. Then we extract semantic representations from the target object description and images to construct an Object Mask Field, which combines with the original image perceptions and robot skill, jointly optimizing the policy. This allows the model to learn the mutual relationships between the target objects identified by the Object Mask Fields, robot skills, and expert actions. Once the target object in the instruction is translated into the Object Mask Field, the policy only needs to interpret the positional information indicated in the Object Mask Field correctly and know how to use the learned robot skills to physically manipulate new target objects, without the need to ground the target object in the instruction beforehand. This formulation enables our system to employ the acquired robotic skills to achieve zero-shot generalization to unseen objects and complete compositional tasks without further fine-tuning (see Figure 1). To summarise, our main contributions are as follows:

- We introduce a novel framework called "Attention before Manipulation" (**ABM**) for generalization in robot manipulation tasks, which can generalize to unseen objects and instructions without further fine-tuning.
- To decouple visual grounding from action prediction, we design an Object Mask Field, which indicates the spatial location of the target object in the point cloud of current scene, serving as a better representation of the target object to assist in policy learning.
- Extensive experiments conducted in both simulation and real world demonstrate that ABM significantly outperforms the baselines in zero-shot and compositional generalization experiment settings. Visual results are provided at: ABM.github.io.

2 Related Work

2.1 Language Conditioned Imitation Learning for Robot Manipulation

Behavior cloning provides a simple and effective approach for robots to learn new skills by mapping observations to expert actions through supervised learning. It enables agents to quickly and efficiently learn from expert actions, making it a widely used imitation learning method in robotics. Models based on behavior cloning are typically trained end-to-end. Recently, an increasing number of studies have integrated human instructions with behavior cloning, allowing robots to execute actions based on human commands. Some of these works leverage pre-trained language models [Brohan *et al.*, 2022; Jiang *et al.*, 2023] or language encoders from pre-trained VLMs [Shridhar *et al.*, 2022; Shridhar *et al.*, 2023; Goyal *et al.*, 2023] to encode instructions, using text features as one of the inputs for agent training. For instance, approaches [Shridhar *et al.*, 2023; Goyal *et al.*, 2023] combine text features encoded by the CLIP language encoder with perceptual features, learning their mutual relationships through an attention module [Vaswani *et al.*, 2017] to jointly optimize the policy. Cliport [Shridhar *et al.*, 2022] , building upon the Transporter manipulation framework [Zeng *et al.*, 2021], integrates pre-trained image and language representations from CLIP to enhance the performance of the original model with a certain degree of generalization capability. However, this approach binds the relationships between perceptions, instructions, and actions at the feature level, limiting the model's generalization ability. In contrast to these works, we use Object Mask Fields to represent the target objects in language instructions, eliminating the agent's reliance on the descriptions of the target object in the instructions. This allows the model to focus more on learning robot skills from instructions, enabling it to generalize the same operational skills to new semantic object categories.

2.2 Foundation Models for Robotics

Foundation models typically refer to models trained on Internet-scale datasets. Typically, we fine-tune pre-trained foundational models for diverse downstream tasks or utilize embedded commonsense to aid downstream models in achieving varied generalization objectives. Recent studies leveraging foundation models demonstrate remarkable generalization and zero-shot capabilities. Instruct2Act [Huang *et al.*, 2023b] uses CLIP image-level features for target retrieval and employs Large Language Models (LLMs) to generate executable Python code for task completion, while Vosposer [Huang *et al.*, 2023c] utilizes LLMs and VLMs to create 3D value maps as objective functions for synthesizing trajectories in a zero-shot manner. VLMap [Huang *et al.*, 2023a] utilizes LSeg [Li *et al.*, 2022] to build a 2D semantic map for navigation, while CLIP-Fields [Shafiullah *et al.*, 2022] learns a mapping from spatial locations to semantic embedding vectors via foundation models. Moreover, several recent works integrate 2D foundation models with 3D feature fields for robotic manipulation. Act3D [Gervet *et al.*, 2023] learns 3D scene feature fields through recurrent coarse-to-fine 3D point sampling and featurization utilizing relative-position attentions. F3rm [Shen *et al.*, 2023] extracts patch-level dense features for the images from CLIP and distills them into a feature field by modeling a NeRF [Mildenhall *et al.*, 2021] for generalization in robot tasks. D^3 Fields [Wang *et al.*, 2023b] utilizes point clouds and foundation models to construct unified descriptor fields that are 3D, dynamic,

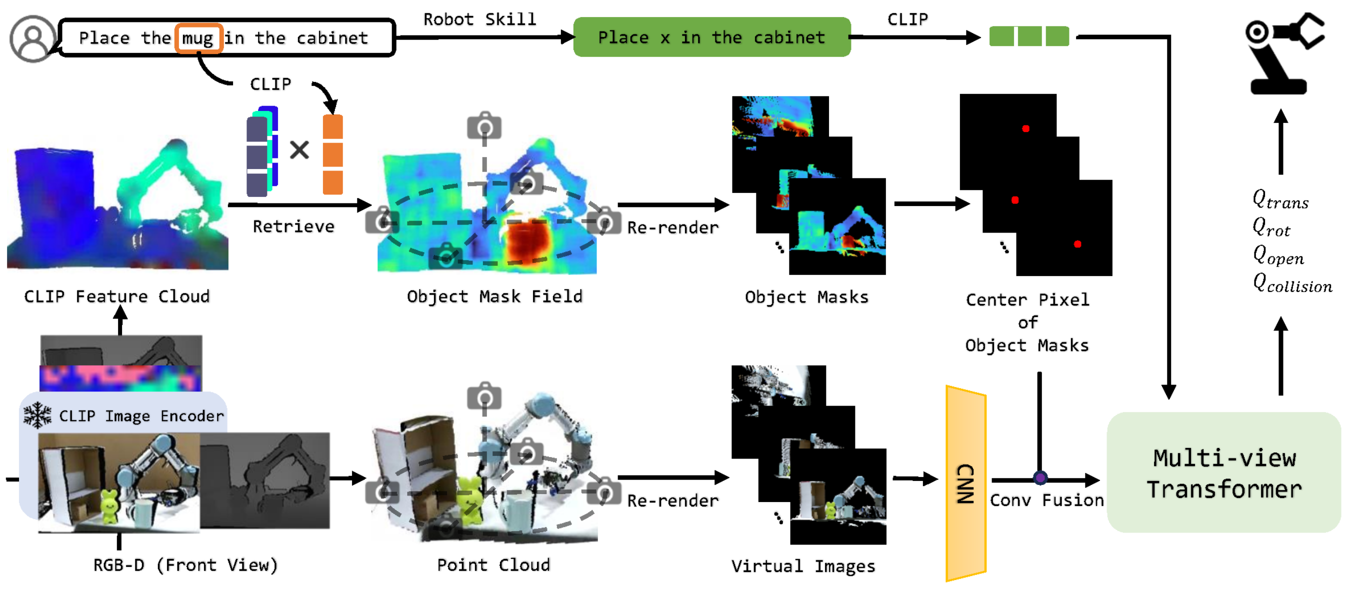

Figure 2: Overview of the proposed ABM. (1) Extract dense features for the image from CLIP backbone and project them in 3D to construct CLIP Feature Cloud (PCA shown). (2) Query CLIP Feature Cloud with target object to generate Object Mask Field (heatmap shown) and re-render to produce center pixel obejct masks. (3) Feed them to a multi-view transformer for actions inference after fusing with virtual images.

and semantic. Inspired by these works, our method similarly constructs a dynamic 3D feature field in the form of point cloud, where each point's features are derived from interpolated patch-level image representations from CLIP, providing strong spatial and semantic prior for downstream robotic manipulation.

2.3 Generalization in Robot Learning

Recent researches focus on enabling agents to generalize physical operations to unseen objects and novel instructions [Jang *et al.*, 2022; Wang *et al.*, 2023a; Jiang *et al.*, 2023]. Furthermore, some studies investigate completing tasks without any specific training by leveraging pre-trained VLMs [Huang *et al.*, 2023c; Huang *et al.*, 2023b]. Programport [Wang *et al.*, 2023a] decouples general visual grounding from policy learning by utilizing the semantic structure of instructions and VLMs, resulting in better generalization. MOO [Stone *et al.*, 2023] trains the policy with masks centered on the detected objects to achieve novel concept grounding, while VIMA [Jiang *et al.*, 2023] achieves this through a novel multimodal prompting formulation. Our approach similarly emphasizes making the policy strongly dependent on visual grounding results, instead of training directly with features from VLMs.

3 Method

We present ABM, whose key idea is to extract rich commonsense from a frozen pre-trained CLIP to construct an Object Mask Field and use it as a better representation of the target object for policy learning. Our objective is to empower the agent to successfully execute manipulation tasks involving novel object categories not included in the training dataset, guided by instructions. An overview of the proposed ABM is shown in Figure 2.

3.1 Demonstrations

We assume that we are given a dataset $D = \{D_1, D_2, \dots, D_n\}$ of n expert demonstrations covering various tasks. Each demonstration corresponds to a sequence of observation-action pairs and an English instruction, *i.e.*, $D_i = \{(o_1, a_1)_i, (o_2, a_2)_i, \dots, (o_t, a_t)_i, l_i\}$. Here, $\{o_1, o_2, \dots, o_t\}_i$ denotes RGB-D image(s) and gripper state at each timestep during a successful roll-out, *i.e.*, $o_t = \{x, a_{\text{open}}\}_t$. $\{a_1, a_2, \dots, a_t\}_i$ denotes the corresponding expert actions, which consist of the 6-DoF pose, gripper open state, and whether the motion planner used collision avoidance to reach an intermediate pose, *i.e.*, $a_t = \{\{a_{trans}, a_{rot}\}_{pose}, a_{open}, a_{collide}\}_t$. l_i represents the language instruction annotating the ongoing task.

3.2 Instruction Decomposition Module

To break the deep binding between instructions, perceptions, and actions in end-to-end training, we isolate the target object description from the training process. To achieve this and inspired by MOO [Stone *et al.*, 2023], we propose an instruction decomposition module to split language instructions. Given an instruction, we first break down instruction into the target object description, representing the object that the robot needs to manipulate in the current episode, and the robot skill, describing how the robot should manipulate the target object. We refer the description of target object as X and robot skill as T_{skill}. We use simple regular expressions Re to split the instructions: $Re(l_i) = \{X, T_{skill}\}_i$. We then use robot skill as the new language goal for the current episode, *i.e.*, $l_i = T^i_{skill}$. For example, for the instruction l_i = "put the tomato in the plate", where T^i_{skill} = "put X in the plate" and X_i = "tomato", the new language goal for the current episode becomes "put X in the plate", *i.e.*, l_i = "put X in the plate". After splitting the instruction, we use the text encoder $\mathcal{H}$ from a pre-trained CLIP model to encode X with the prompt "a photo of a(an) X", refered as X_{prompt}, to obtain text features $\mathcal{H}(X_{prompt}) \in R^{1\times D}$, which will be used to retrieve the area-of-interest in the CLIP Feature Cloud described in the next subsection. We denote the text feature dimensions as D. Then, we utilize text encoder $\mathcal{H}$ to encode robot skill T_{skill}, extracting the values of the text encoder's last layer before projection as token embeddings of new language goal for policy learning, $\mathcal{H}(T_{skill}) \in R^{77\times D}$.

The purpose is to enable the policy to learn the abstract correspondence between the character X and the rough 3D spatial position of the target object identified by the Object Mask Field described in the next subsection. We can also query large language models (LLMs) such as ChatGPT [Ouyang *et al.*, 2022] to parse instructions, standardizing free-form user inputs into the specified template.

3.3 Object Mask Field for Robotic Manipulation

To illustrate the concept, we only consider an RGB-D image as an example, and for multiple camera inputs, just simply repeat the operation and stack them together. Given the RGB image $x_t \in R^{H\times W\times 3}$, we use the image encoder $\mathcal{G}$ from a pre-trained CLIP model to extract patch-level dense features from the penultimate layer of $\mathcal{G}$ according to the MaskCLIP reparameterization trick [Zhou *et al.*, 2022], $\mathcal{G}(x_t) \in R^{H'\times W'\times D}$. Here, we denote the spatial dimensions of visual features as $H' \times W'$ and the feature dimensions as D. Additionally, the extracted patch-level dense features still maintain alignment with the CLIP text features in the same feature space to support language guidance in subsequent processing.

To project patch-level dense features into 3D space, we restore the patch-level 2D feature map to the same resolution as the original image by interpolation, *i.e.*, $\mathcal{G}(x_t) \in R^{H'\times W'\times D} \rightarrow \mathcal{G}'(x_t) \in R^{H\times W\times D}$. Therefore, we obtain a pixel-resolution dense feature map interpolated from the patch-level 2D feature map, $\mathcal{G}'(x_t) = \mathcal{I}n(\mathcal{G}(x_t))$, where $\mathcal{I}n(\cdot)$ denotes an interpolation operator. And to preserve the integrity of CLIP image features to the maximum extent, we fill in with the nearest patch features based on relative positions rather than using linear interpolation.

Then we reconstruct a point cloud of the scene through RGB-D image and project the obtained pixel-resolution dense feature map, *i.e.* $\mathcal{G}'(x_t)$, onto the corresponding points of the point cloud via sensed depth, forming a semantic point cloud, which we refer as CLIP Feature Cloud $\mathcal{F} \in R^{N\times D}$, where N denotes the number of 3D points in CLIP Feature Cloud. As CLIP aligns image features and text features in a shared feature space, we query the CLIP Feature Cloud $\mathcal{F}$ with the target object text features $\mathcal{H}(X_{prompt})$ to generate heatmap $\mathcal{HF} \in R^{N\times 1}$ through cosine similarity calculation, where we aim for the points of the target object to have higher values in the heatmap. Here, $\mathcal{HF}$ is defined as $\mathcal{HF} = \mathcal{F} \otimes \mathcal{H}(X_{prompt})$. For ease of training, we use an one-hot single channel Object Mask Field $\mathcal{M} \in R^{N\times 1}$ to simplify the heatmap. The values are set to 1.0 where the heatmap values are greater than the threshold we set manually; otherwise, we set them to 0 in $\mathcal{M}$. Here, we set the threshold to **0.7**, *i.e.*, $\mathcal{M}(p) = 1.0$, where $\mathcal{HF}(p) > 0.7$, otherwise $\mathcal{M}(p) = 0$. Here, p denotes an arbitrary 3D point in $\mathcal{M}$ and $\mathcal{HF}$. Therefore, we obtain an Object Mask Field for robotic manipulation during training and evaluation.

This design leverages rich commonsense from the pre-trained CLIP to indicate the rough position of the target object in the workspace, providing the policy with a better representation of the target object. Importantly, we build the Object Mask Field with a frozen CLIP model so that it does not specialize or overfit to the objects in the demonstrations, and the policy can learn to be robust to errors made by CLIP during the training loop.

3.4 Architecture and Training for Robotic Policy

In our architecture, our policy model extends the architecture from RVT [Goyal *et al.*, 2023] with following modifications.

First, as mentioned above in subsection 3.2, instead of using the entire language goal of each task, we use the robot skill T_{skill} as the new language goal, and encode it through text encoder $\mathcal{H}$ to obtain token embeddings $\mathcal{H}(T_{\text{skill}}) \in R^{77\times D}$ for policy learning.

Second, we re-render Object Mask Field $\mathcal{M}$ and point cloud of the scene by Pytorch3D [Ravi *et al.*, 2020] to obtain four maps with a total of 8 channels in each virtual image. Virtual images are re-rendered from five virtual viewpoints: front, back, left, right, and above of the robot base. We refer virtual images from each viewpoint as $\mathcal{V}_i \in R^{220\times 220\times 8}$, where $i \in \{front, back, left, right, above\}$. And the four maps respectively represent object mask ($Map_1 \in R^{220\times 220\times 1}$), RGB ($Map_2 \in R^{220\times 220\times 3}$), depth ($Map_3 \in R^{220\times 220\times 1}$), and (x, y, z) coordinates of the points in the world frame ($Map_4 \in R^{220\times 220\times 3}$). The last three maps are the same as RVT, and we refer them as $\mathcal{V}'_i = \{Map_2, Map_3, Map_4\}_i \in R^{220\times 220\times 7}$. Specially, object mask indicates the rough position of the target object in the virtual image. Since object mask is re-rendered from the Object Mask Field, which indicates the target object's position in 3D space with limited accuracy, we post-process the object mask by taking the central pixel coordinate of non-zero region in the object mask as the target object's position in the virtual image. In other words, object mask becomes $Map'_1(Center(\{Map_1(u,v) \neq 0\})) = 1.0$, otherwise $Map'_1(u,v) = 0$, where (u,v) denotes an arbitrary pixel coordinate of the virtual image and $Center(\cdot)$ denotes center pixel calculation. (see Figure 3 (b) center-pixel mask for example).

Third, we expect that the policy should be conditioned on the object masks mentioned above (*i.e.*, five spatial object masks indicating the position of the target object). Therefore, we feed them into the joint transformer architecture of RVT with a few modifications. We fuse Map'_1 and $\mathcal{V}'_i$ to obtain perception features and break each of them into 20 × 20 patches to produce image token embeddings:

$$Emb_{image} = \mathcal{F}_{patchify}(\mathcal{F}_{fuse}(Cat(Map'_1, \mathcal{F}_{cnn}(\mathcal{V}'_i)))) \quad (1)$$

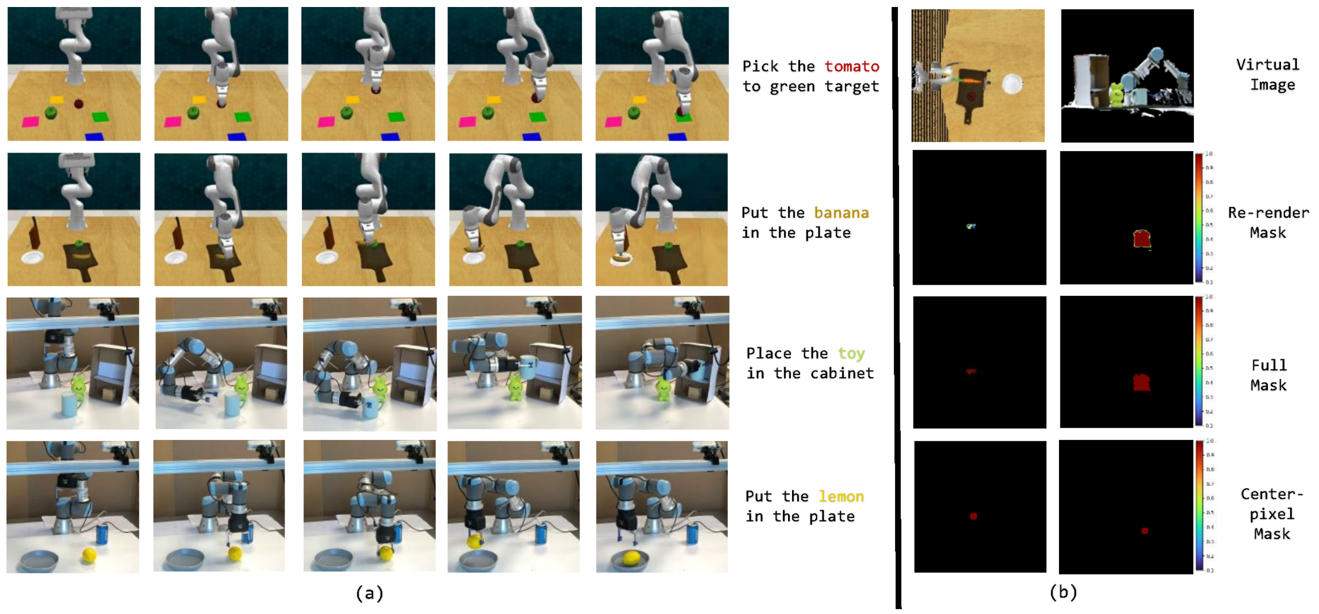
Pick the tomato to green target
Put the banana in the plate
Place the toy in the cabinet
Put the lemon in the plate
(a)
Virtual Image
Re-render Mask
Full Mask
Center-pixel Mask
(b)

Figure 3: (a) Examples of evaluation trajectories across various instructions in simulation and real-world. (b) Examples of object mask type.

where $\mathcal{F}_{fuse}$ denotes a fusion layer, and $\mathcal{F}_{cnn}$ denotes a CNN feature extractor. $\mathcal{F}_{patchify}$ denotes a CNN layer to patchify the fused features and Cat denotes concatenation operator. Then Emb_{image} is fed to eight transformers for the action decoder, which predicts the target end effector pose (Q_{trans}, Q_{rot}), gripper state (Q_{open}), and collision ($Q_{\text{collision}}$). We refer readers to the RVT paper for more details about the architecture.

3.5 Training

We train our model end-to-end through behavior cloning by minimizing the negative log-likelihood of predicted actions. Concretely, given observation-action tuples (o_t, l_t, a_t) sampled from expert demonstrations, our goal is to minimize:

$$\min_{\theta} \sum_{(o_t,l_t,a_t)\sim D} -\log \pi_\theta(a_t|o_t, l_t) \tag{2}$$

where t represents a certain timestep within the episode. We use the loss function following RVT [Goyal *et al.*, 2023].

4 Experiments

4.1 Simulation Experiments

Simulation Setup

Environment. Following to the simulation configuration of RVT, we utilize CoppelaSim [Rohmer *et al.*, 2013] to simulate a range of tasks from RLBench [James *et al.*, 2020] which is interfaced through PyRep [James *et al.*, 2019]. All tasks are executed by controlling a Franka Panda robot equipped with a parallel gripper at the center of dinner table. Visual observations are captured from four noiseless RGB-D cameras located at the front, left shoulder, right shoulder, and on the wrist, each with a resolution of 128 × 128. Given the target end effector pose, the Franka Panda robot will reach target through motion planner.

RLBench Tasks. We train our model on 8 RLBench tasks, where 6 tasks are adapted from the training tasks in RVT with slight modifications to meet the experimental requirements. Additionally, two tasks are entirely new tasks created using the RLBench tool. Each task includes several variations with corresponding language instructions. For example, in the "place food in plate" task, instructions like "put the carrot in the plate" and "put the tomato in the plate" represent two different variations of the task. These variations are randomly sampled during the generation of the training dataset. However, unlike RVT, to assess the zero-shot generalization performance of the model during evaluation, we make corresponding modifications to the validation sets of these 8 tasks, which include colors, object quantities, and object categories that have never been seen before. Of course, the agent also needs to manage new object poses and goals, which are randomly sampled, similar to RVT. In addition, we create 2 extra compositional tasks to assess the model's ability to combine learned robot skills with new object categories. These tasks involve scenarios composed of combinations of two different training tasks.

Baselines. The efficacy of our proposed method is assessed by comparing it with two baselines: Robotic View Transformer (RVT) and Integrate Clip Features Directly (IFD). RVT is a multi-view transformer designed to predict actions by fusing images re-rendered within the robot's workspace. IFD, a modification of RVT, differs in that it re-renders not only virtual images around the robot but also clip feature maps corresponding to these images. IFD directly integrates these clip feature maps with the virtual images and incorporates language goals for action prediction.

Training and Evaluation Details. We utilize RLBench tools to generate training datasets with 100 demonstrations per task as RVT. For ABM data preprocessing, we employ the

Method		Avg. Success	Put in Safe	Place Wine	Water Plants	Close Jar	Insert Peg	Meat off Grill	Pick Block	Place Food
RVT [Goyal *et al.*, 2023]	seen	**74.2**	86.4	87.2	48	**56**	**25.6**	**91.2**	**100**	**99.2**
IFD	seen	70.9	**90.4**	**89.6**	54.4	46.4	9.6	83.2	99.2	94.4
ABM (Ours)	seen	70	86.4	87.2	**55.2**	48.8	14.4	75.2	97.6	95.2
RVT [Goyal *et al.*, 2023]	unseen	24	19.2	5.6	20	**33.6**	5.6	52	35.2	20.8
IFD	unseen	34.5	0.8	15.2	41.6	**33.6**	11.2	92.8	54.4	26.4
ABM (Ours)	unseen	**57.5**	**52.8**	**51.2**	**43.2**	28	**16**	**95.2**	**90.4**	**83.2**

Table 1: Multi-Task Test Result. "unseen" represents the model's performance in zero-shot generalization validation set, where the objects are unseen during the training process. "seen" represents the model's performance in the original validation set, where the objects are seen during the training process.

frozen, pretrained ViT-L/14@336px CLIP encoder to encode the images from four RGB-D cameras, and obtain patch-level dense features, which will be upscaled to match the resolution of the original images captured by the cameras during training. We use a batch size of 30 to train our model and baseline methods on 6 NVIDIA RTX 3090 GPUs for 80k iterations with the LAMB optimizer [You *et al.*, 2019] and a learning rate of 0.003. During training, data augmentation involves random translations of point clouds within the range of [± 0.125m, ± 0.125m, ± 0.125m], and random rotations around the yaw axis within the range of ± 45°. Evaluations are scored as 0 for failures or 100 for complete successes, and we report average success rates by evaluating the model five times on the same 25 variations episodes per task in "seen", "unseen" and compositional generalization evaluations.

Zero-short Generalization Performance

To validate the zero-shot generalization performance of the ABM, we separate the objects used in the training set from those in the validation set. Specifically, there is no overlap between the objects in the training and validation sets, meaning that the objects in the validation set have never been seen by the model during the training process.

Table 1 compares the zero-shot generalization performance between ABM and the baselines. We find that ABM achieves similar average success rates to the baselines on all tasks in the "seen" validation set, which are 4.2% lower than RVT and 0.9% lower than IFD. However, in the "unseen" validation set, ABM significantly outperforms the two baseline methods in 87.5% (7/8) of tasks, with average success rates 33.5% higher than RVT and 23% higher than IFD, respectively. Overall, experimental results demonstrate that ABM exhibits stronger zero-shot generalization ability compared to the baseline methods, while IFD, which integrates clip features into policy optimization directly, achieves only limited generalization.

Compositional Generalization Performance

To assess the model's ability to generalize in compositional tasks, we design two completely new tasks. Instead of training on these tasks, we evaluate the performance of our model and the baselines directly in the environments of these two tasks. The concepts of the skills required for these tasks originate from two training tasks, and successfully completing these tasks would necessitate the model's ability to combine the learned robot skills from the training process with target objects from different tasks. For instance, in the "place block in plate" task, "put x in the plate" represents the robot skill learned from "place food in plate" task, and "block" comes from different task named "pick block to color target".

Method	Avg. Success	Pick Food	Place Block
RVT [Goyal *et al.*, 2023]	21.6	20	23.2
IFD	31.2	5.6	56.8
ABM (Ours)	**53.2**	**47.2**	**59.2**

Table 2: Compositional Task Result.

Table 2 compares the compositional generalization performance between ABM and the baselines. We find that the average success rate of ABM in compositional tasks is 31.6% higher than RVT and 22% higher than IFD respectively. The poor performance of baselines in compositional tasks can be attributed to their learning of a mapping from observations to expert actions in the training set, without acquiring specific robot skill for each task. In contrast, ABM learns abstract robot skills from training tasks and combines them with Object Mask Fields of new scenes to generalize learned robot skills to new objects. This, to some extent, demonstrates ability of ABM to reason and combine learned skills with objects indicated by the Object Mask Fields.

Ablation Studies

To investigate the impact of different object mask fusion methods and representations on the model, we conduct additional ablation studies. It is worth noting that in the Object Mask Field, the value of each 3D point corresponding to the target object's position is 1.0, while the rest are 0.0. However, the value of object mask re-rendered by Pytorch3D [Ravi *et al.*, 2020] at each point in the target object's position ranges from 0.0 to 1.0, and we refer to this as the "re-rendered mask" (see Figure 3 (b) re-rendered mask). Setting all non-zero values in the "re-rendered mask" to 1.0, we refer to this object mask as the "full mask" (see Figure 3 (b) full mask).

Method		Avg. Success	Put in Safe	Place Wine	Water Plants	Close Jar	Insert Peg	Meat off Grill	Pick Block	Place Food
Stack in channel	seen	**72.8**	87.2	93.6	48	**58.4**	14.4	**87.2**	95.2	98.4
Concatenate with re-rendered mask	seen	72.6	**92.8**	89.6	45.6	52	**24.8**	84.8	92.8	98.4
Concatenate with full mask	seen	71.9	81.6	**94.4**	51.2	52	16	85.6	95.2	**99.2**
Concatenate with center-pixel mask (*ABM*)	seen	70	86.4	87.2	**55.2**	48.8	14.4	75.2	**97.6**	95.2
Stack in channel	unseen	54.9	41.6	48.8	33.6	40.8	8	92	**99.2**	75.2
Concatenate with re-rendered mask	unseen	54.7	37.6	45.6	13.6	52.8	**16**	89.6	96	**86.4**
Concatenate with full mask	unseen	57	27.2	**67.2**	37.6	**58.4**	12	80	97.6	76
Concatenate with center-pixel mask (*ABM*)	unseen	**57.5**	**52.8**	51.2	**43.2**	28	**16**	**95.2**	90.4	83.2

Table 3: Ablation studies of proposed method with different object mask fusion methods and representations.

The results of the ablation experiments are reported in Table 3. In this context, "stack in channel" means that we append the re-rendered masks channel-wise to the virtual images and feed them directly to the policy. And "concatenate with re-rendered mask" means that we concatenate the "re-rendered mask" with the visual features, which are obtained by feeding the virtual images to preprocessing MLP. Similarly, "concatenate with full mask" involves concatenating the "full mask" with the visual features. In summary, the approach of concatenating object masks with the policy model yields better results in "unseen" setting. Moreover, indicating the object's position in masks with a single pixel, as done by ABM, performs better than other representations. This may be because the Object Mask Field is obtained from imprecise and unstable segmentation, and using inaccurate masks like "re-rendered mask" and "full mask" may confuse the model.

4.2 Real-World Experiments

Real-World Setup

Hardware. In real-world experiments, we employ a UR3 robot equipped with a parallel gripper for manipulation and a RealSense D435i RGB-D camera mounted on top, pointing at the workspace for perception. The camera provides RGB-D images with a resolution of 1280×720 at 30 Hz. Before being fed to the model, point clouds are transformed to the robot base frame using the extrinsics between the robot and the camera.

Real-World Tasks. We train our ABM model from scratch on two tasks (with 4 variations), using a total of 40 demonstrations. For each task evaluation, we use 10 task scenarios for in-distribution target objects and another 10 for out-of-distribution target objects. Similar to the simulation environment, each task is described by language instructions specifying different variations.

Result

We train our ABM model on a real-world dataset using a configuration similar to the simulation. We train the agent for 20k iterations with a learning rate of 0.0032. Evaluation results are reported in Table 4. Overall, our model achieve average success rates of 55% in the "seen" test tasks and 65% in the "unseen" test tasks. Note that "unseen" has a higher average success rate than "seen" on real-world tasks, and we believe that the insufficient number of validations is the cause.

Method		Avg. Success	Place in Plate	Place in Cabinet
ABM	seen	55	40	70
ABM	unseen	65	50	80

Table 4: Real-world Task Result.

For each task in both the "seen" and "unseen" settings, we only conduct validations on 10 scenes, leading to potential randomness in the validation results. Conducting multiple rounds of validation on a larger number of scenes and averaging the results would mitigate this issue. We assess the real-time performance of our model on an NVIDIA RTX 3090, achieving inference speeds of approximately 0.68 FPS for the model and 0.98 FPS for generating object masks. Employing keypoint tracking as mentioned in the limitations to track the target instance from the initial timestep can significantly enhance the system's inference speed.

5 Conclusions and Limitations

We present ABM, a transformer-based model, which leverages rich commonsense from pre-trained CLIP for downstream policy learning. Simulation and real-world experiments demonstrate the impressive generalization ability of ABM, enabling agent to complete tasks which contain novel instructions and objects that are never seen before.

While ABM achieves promising results, there are still some important limitations: (1) ABM requires constructing the Object Mask Field at each timestep, leading to varying spatial positions of the identified target object at each timestep and time-consuming, which are detrimental to policy optimization. This issue can be addressed by employing keypoint tracking to track the target instance from the first timestep and we don't need to construct the Object Mask Field in later timesteps. (2) Currently, our method can only focus on a single target object in a task and apply learned skills to that object, limiting its application to more complex multi-target object manipulation tasks. We can solve this problem by decomposing the language goals into descriptions of multiple subtasks, and focusing on the target objects specified by the corresponding descriptions at different timesteps.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62271324, 62231020, 62371309 and 61976142, Shenzhen Science and Technology Program under Grant ZDSYS20220527171400002.

References

[Brohan *et al.*, 2022] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, *et al.* RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[Gervet *et al.*, 2023] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3D: 3D Feature Field Transformers for Multi-Task Robotic Manipulation. In *Proceedings of the Conference on Robot Learning*, pages 3949–3965, 2023.

[Goyal *et al.*, 2023] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic View Transformer for 3D Object Manipulation. *arXiv preprint arXiv:2306.14896*, 2023.

[Huang *et al.*, 2023a] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615, 2023.

[Huang *et al.*, 2023b] Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model. *arXiv preprint arXiv:2305.11176*, 2023.

[Huang *et al.*, 2023c] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[James *et al.*, 2019] Stephen James, Marc Freese, and Andrew J Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[James *et al.*, 2020] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *IEEE Robotics and Automation Letters*, pages 3019—-3026, 2020.

[Jang *et al.*, 2022] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *Proceedings of the Conference on Robot Learning*, pages 991–1002, 2022.

[Jiang *et al.*, 2023] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: General Robot Manipulation with Multimodal Prompts. In *Proceedings of the Fortieth International Conference on Machine Learning*, 2023.

[Li *et al.*, 2022] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven Semantic Segmentation. In *Proceedings of the International Conference on Learning Representations*, 2022.

[Liu *et al.*, 2022] Hao Liu, Lisa Lee, Kimin Lee, and Pieter Abbeel. Instruction-following agents with jointly pre-trained vision-language models. *arXiv preprint arXiv:2210.13431*, 2022.

[Mildenhall *et al.*, 2021] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[Ouyang *et al.*, 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, *et al.* Training language models to follow instructions with human feedback. *Proceedings of the Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[Paradis *et al.*, 2021] Samuel Paradis, Minho Hwang, Brijen Thananjeyan, Jeffrey Ichnowski, Daniel Seita, Danyal Fer, Thomas Low, Joseph E Gonzalez, and Ken Goldberg. Intermittent visual servoing: Efficiently learning policies robust to instrument changes for high-precision surgical manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7166–7173, 2021.

[Rahmatizadeh *et al.*, 2018] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765, 2018.

[Ravi *et al.*, 2020] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020.

[Rohmer *et al.*, 2013] Eric Rohmer, Surya PN Singh, and Marc Freese. V-REP: A versatile and scalable robot simulation framework. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326, 2013.

[Shafiullah *et al.*, 2022] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022.

[Shen *et al.*, 2023] William Shen, Ge Yang, Alan Yu, Jansen Wong, Leslie Pack Kaelbling, and Phillip Isola. Distilled feature fields enable few-shot language-guided manipulation. *arXiv preprint arXiv:2308.07931*, 2023.

[Shridhar *et al.*, 2022] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the Conference on Robot Learning*, pages 894–906, 2022.

[Shridhar *et al.*, 2023] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the Conference on Robot Learning*, pages 785–799, 2023.

[Stepputtis *et al.*, 2020] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

[Stone *et al.*, 2023] Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Brianna Zitkovich, Fei Xia, Chelsea Finn, *et al.* Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proceedings of the Advances in Neural Information Processing Systems*, 30, 2017.

[Wang *et al.*, 2023a] Renhao Wang, Jiayuan Mao, Joy Hsu, Hang Zhao, Jiajun Wu, and Yang Gao. Programmatically Grounded, Compositionally Generalizable Robotic Manipulation. *arXiv preprint arXiv:2304.13826*, 2023.

[Wang *et al.*, 2023b] Yixuan Wang, Zhuoran Li, Mingtong Zhang, Katherine Driggs-Campbell, Jiajun Wu, Li Fei-Fei, and Yunzhu Li. D˜3 Fields: Dynamic 3D Descriptor Fields for Zero-Shot Generalizable Robotic Manipulation. *arXiv preprint arXiv:2309.16118*, 2023.

[You *et al.*, 2019] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

[Zeng *et al.*, 2021] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, *et al.* Transporter networks: Rearranging the visual world for robotic manipulation. In *Proceedings of the Conference on Robot Learning*, pages 726–747, 2021.

[Zhang *et al.*, 2018] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635, 2018.

[Zhou *et al.*, 2022] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *Proceedings of the European Conference on Computer Vision*, pages 696–712, 2022.