PERCEIVER-ACTOR: A Multi-Task Transformer for Robotic Manipulation

Anonymous Author(s)
Affiliation
Address
email

peract.github.io

Abstract: Transformers have revolutionized vision and natural language processing with their ability to scale with large datasets. But in robotic manipulation, data is both limited and expensive. Can we still benefit from Transformers with the right problem formulation? We investigate this question with PERACT, a language-conditioned behavior-cloning agent for multi-task 6-DoF manipulation. PERACT encodes language goals and RGB-D voxel observations with a Perceiver Transformer [1], and outputs discretized actions by "detecting the next best voxel action". Unlike frameworks that operate on 2D images, the voxelized observation and action space provides a strong structural prior for efficiently learning 6-DoF policies. With this formulation, we train a single multi-task Transformer for 18 RLBench and 7 real-world tasks with just a few demonstrations per task. Our results show that PERACT significantly outperforms unstructured image-to-action agents and 3D ConvNet baselines for a wide range of tabletop tasks.

Keywords: Transformers, Language Grounding, Manipulation, Behavior Cloning

1 Introduction

Transformers [2] have become prevalent in natural language processing and computer vision. By framing problems as sequence modeling tasks, and training on large amounts of diverse data, Transformers have achieved groundbreaking results in several domains [3, 4, 5, 6]. Even in domains that do not conventionally involve sequence modeling [7, 8], Transformers have been adopted as a generalist architecture [9]. But in robotic manipulation, data is both limited and expensive. Can we still bring the power of Transformers to 6-DoF manipulation with the right problem formulation?

Language models operate on sequences of tokens [10], and vision transformers operate on sequences of image patches [4]. While pixel transformers [11, 1] exist, they are not as data efficient as approaches that use convolutions or patches to exploit the 2D structure of images. Thus, while Transformers may be domain agnostic, they still require the right problem formulation to be data efficient. A similar efficiency issue is apparent in behavior-cloning (BC) approaches that directly train image-to-action agents for 6-DoF manipulation. In Gato [9], a multi-task Transformer was trained on several domains, including MetaWorld tasks [12] like opening a drawer and pressing a button, but this required a total of 95K episodes. Relatedly, recent works in language-conditioned behavior-cloning [13, 14] for manipulation have shown impressive multi-task capabilities, but they require several days or even months of data collection. Instead of collecting thousands of demonstrations, in this work, we aim to exploit the 3D structure of voxelized observations for efficient 6-DoF behavior cloning (analogous to how vision transformers exploit the 2D structure of images with patches).

To this end, we present PERACT (short for PERCEIVER-ACTOR), a language-conditioned BC agent that can learn to imitate a wide variety of 6-DoF manipulation tasks with just a few demonstrations per task. Instead of operating on image patches, PERACT encodes a sequence of RGB-D voxel patches and predicts discretized translations, rotations, and gripper actions that are executed with

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

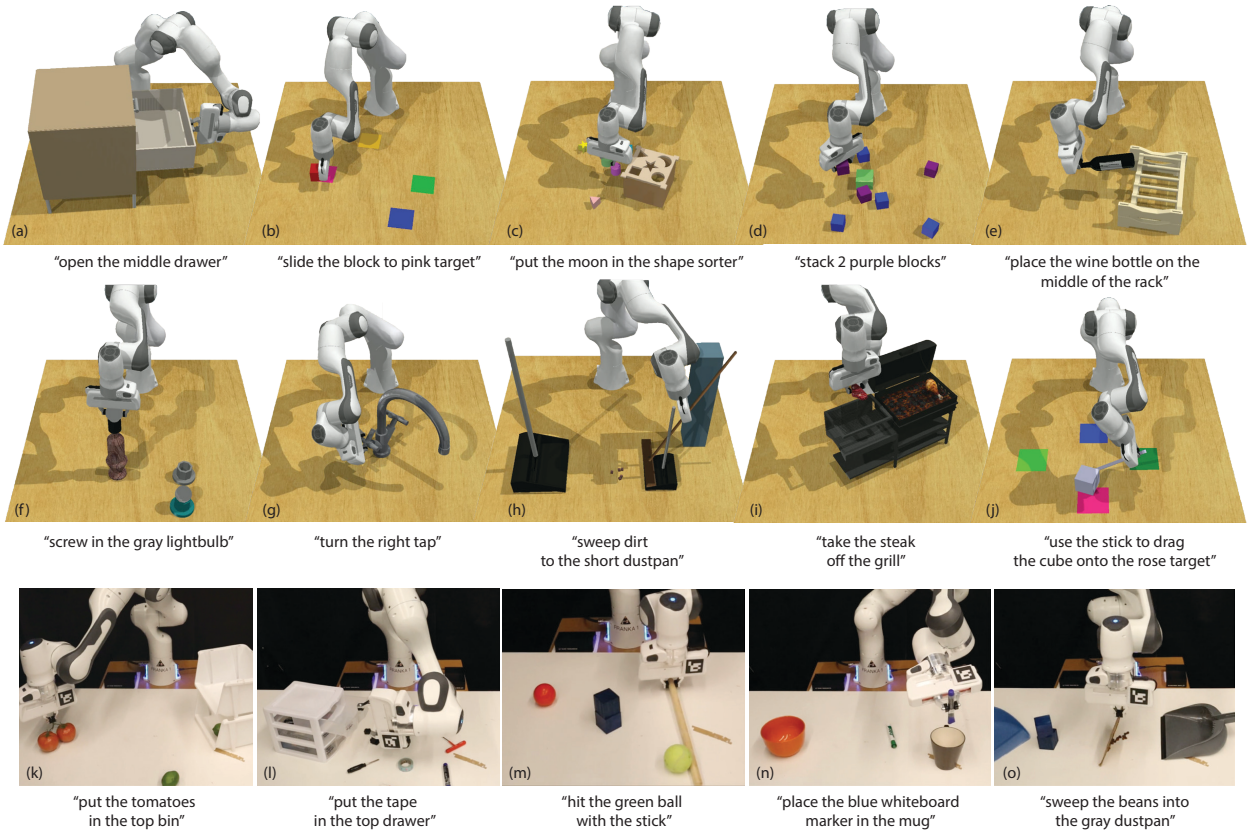

Figure 1. **Language-Conditioned Manipulation Tasks:** PERACT is a language-conditioned multi-task agent capable of imitating a wide range of 6-DoF manipulation tasks. We conduct experiments with 18 simulated tasks in RLBench [15] (a-j; only 10 shown), each with several pose and semantic variations. We also demonstrate our approach with a Franka Panda on 7 real-world tasks (k-o; only 5 shown) with a multi-task agent trained with just 53 demonstrations. See the supplementary video for simulated and real-world rollouts.

a motion-planner in an observe-act loop. PERACT is essentially a classifier trained with supervised learning to *detect actions* akin to prior work like CLIPort [16], except our observations and actions are represented with 3D voxels instead of 2D pixels. Voxel grids are less prevalent than images in end-to-end BC approaches often due to scaling issues with high-dimensional inputs. But in PERACT, we use a Perceiver[1] Transformer [1] to encode very high-dimensional input of up to 100^3 voxels with only a small set of latent vectors. This voxel-based formulation provides a strong structural prior with several benefits: a natural method for fusing multi-view observations, learning robust action-centric representations [17], and enabling data augmentation in 6-DoF – all of which help learn generalizable skills by focusing on *diverse* rather than narrow multi-task data.

To study the effectiveness of our formulation, we conduct large-scale experiments in the RLBench [15] environment. We train a single multi-task agent on 18 diverse tasks that involve a range of prehensile and non-prehensile behaviors like placing wine bottles on a rack and dragging objects with a stick (see Figure 1 a-j). Each task also includes several pose and semantic variations with objects that differ in placement, color, shape, size, and category. Our results show that PERACT significantly outperforms unstructured image-to-action agents and 3D ConvNet baselines, without using any explicit representations of instance segmentations, object poses, memory, or symbolic states. We also validate our approach on a Franka Panda with a multi-task agent trained from scratch on 7 real-world tasks with a **total of just 53 demonstrations** (see Figure 1 k-o).

In summary, our contributions are as follows:

- A **novel problem formulation** for perceiving, acting, and specifying goals with Transformers.
- An **action-centric** framework for **grounding language in 6-DoF** manipulation actions.
- **Empirical results** investigating multi-task agents on a range of simulated and real-world tasks.

The code and pre-trained models will be available upon publication at `peract.github.io`.

[1]Throughout the paper we refer to PerceiverIO [1] as Perceiver for brevity.

2 Related Work

Vision for Manipulation. Traditionally, methods in robot perception have used explicit "object" representations like instance segmentations, object classes, poses [18, 19, 20, 21, 22, 23]. Such methods struggle with deformable and granular items like cloths and beans that are hard to represent with geometric models or segmentations. In contrast, recent methods [16, 24] learn action-centric representations without any "objectness" assumptions, but they are limited to top-down 2D settings with simple pick-and-place primitives. In 3D, James et al. proposed C2FARM [25], an action-centric reinforcement learning (RL) agent with a coarse-to-fine-grain 3D-UNet backbone. The coarse-to-fine-grain scheme has a limited receptive field that cannot look at the entire scene at the finest level. In contrast, PERACT learns action-centric representations with a global-receptive field through a Transformer backbone. Also, PERACT does BC instead of RL, which enables us to easily train a multi-task agent for several tasks by conditioning it with language goals.

End-to-End Manipulation approaches [26, 27, 28, 29] make the least assumptions about objects and tasks, but are often formulated as an image-to-action prediction task. Training directly on RGB images for 6-DoF tasks is often inefficient, generally requiring several demonstrations or episodes just to learn basic skills like rearranging objects. In contrast, PERACT uses a voxelized observation and action space, which is dramatically more efficient and robust in 6-DoF settings. While other works in 6-DoF grasping [30, 31, 32, 33, 34] have used RGB-D and point cloud input, they have not been applied to sequential tasks or used with language-conditioning. Another line of work tackles data inefficiency by using pre-trained image representations [16, 35, 36] to bootstrap BC. Although our framework is trained from scratch, such pre-training approaches can be integrated together in future works for even greater efficiency and generalization to unseen objects.

Transformers for Agents and Robots. Transformers have become the prevalent architecture in several domains. Starting with NLP [2, 3, 37], recently in vision [4, 38], and even RL [8, 39, 40]. In robotics, Transformers have been applied to assistive teleop [41], legged locomotion [42], path-planning [43, 44], imitation learning [45, 46], and grasping [47]. Transformers have also achieved impressive results in multi-domain settings like in Gato [9] where a single Transformer was trained for 16 domains such as captioning, language-grounding, robotic control etc. However, Gato relies on extremely large datasets like 15K episodes for block stacking and 94K episodes for Meta-World [12] tasks. Our voxel-based approach might complement agents like Gato, which could use our problem formulation for greater efficiency and robustness in 6-DoF manipulation settings.

Language Grounding for Manipulation. Several works have proposed methods for grounding language in robot actions [48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]. However, these methods use disentangled pipelines for perception and action, with the language primarily being used to guide perception. Recently, a number of end-to-end approaches [13, 14, 59] have been proposed for conditioning BC agents with language instructions. These methods require 1000s of human teleoperated demonstrations that are collected over several days or even months. In contrast, PERACT can learn robust multi-task policy with just a few minutes of training data. For benchmarking, several simulation environments exist [60, 24, 12], but we use RLBench [15] for its diversity of 6-DoF tasks and ease of generating expert demonstrations with templated language goals.

3 PERCEIVER-ACTOR

PERACT is a language-conditioned behavior-cloning agent for 6-DoF manipulation. The key idea is to learn perceptual representations of actions conditioned on language goals. Given a voxelized reconstruction of a scene, we use a Perceiver Transformer [1] to learn per-voxel features. Despite the extremely large input space (100^3), Perceiver uses a small set of latent vectors to encode the input. The encoded features are then used to predict the next best action in terms of discretized translation, rotation, and gripper state at each timestep. PERACT relies purely on the current observation to determine what to do next in sequential tasks. See Figure 2 for an overview of PERACT.

Section 3.1 and Section 3.2 describe our dataset setup. Section 3.3 describes our problem formulation for PERACT, and Section 3.4 provides details on training PERACT.

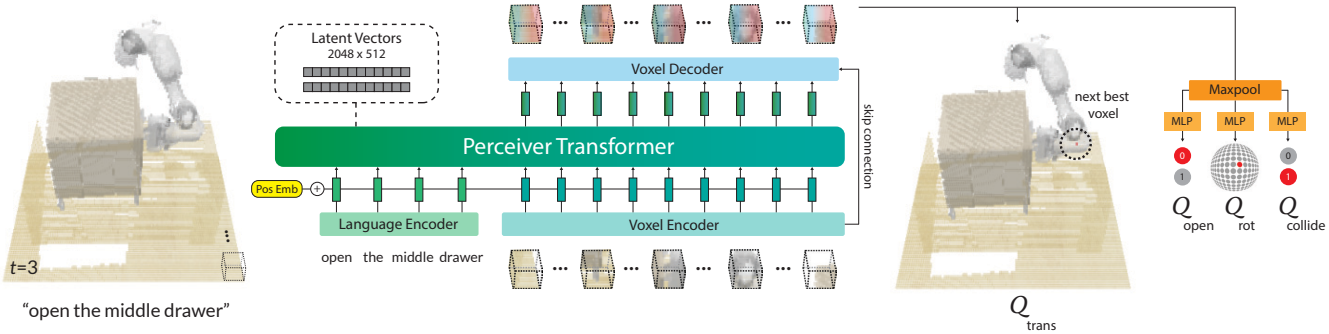

Figure 2. **PERACT Overview.** PERACT is a language-conditioned behavior-cloning agent trained with supervised learning to *detect actions*. PERACT takes as input a language goal and a voxel grid reconstructed from RGB-D sensors. The voxels are split into 3D patches, and the language goal is encoded with a pre-trained language model. These language and voxel features are appended together as a sequence and encoded with a Perceiver transformer [1]. Despite the extremely long input sequence, Perceiver uses a small set of latent vectors to encode the input (see Appendix Figure 6 for an illustration). These encodings are upsampled back to the original voxel dimensions with a decoder and reshaped with linear layers to predict discretized translation, rotation, gripper open, and collision avoidance actions. This action is executed with a motion-planner after which the new observation is used to predict the next discrete action in an observe-act loop until termination.

3.1 Demonstrations

We assume access to a dataset $\mathcal{D} = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$ of n expert demonstrations, each paired with English language goals $\mathcal{G} = \{\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_n\}$. These demonstrations are collected by an expert with the aid of a motion-planner to reach intermediate poses. Each demonstration ζ is a sequence of continuous actions $\mathcal{A} = \{a_1, a_2, \ldots, a_t\}$ paired with observations $\mathcal{O} = \{\tilde{o}_1, \tilde{o}_2, \ldots \tilde{o}_t\}$. An action a consists of the 6-DoF pose, gripper open state, and whether the motion-planner used collision avoidance to reach an intermediate pose: $a = \{a_{\text{pose}}, a_{\text{open}}, a_{\text{collide}}\}$. An observation $\tilde{o}$ consists of RGB-D images from any number of cameras. We use four cameras for simulated experiments $\tilde{o}_{\text{sim}} = \{o_{\text{front}}, o_{\text{left}}, o_{\text{right}}, o_{\text{wrist}}\}$, but just a single camera for real-world experiments $\tilde{o}_{\text{real}} = \{o_{\text{front}}\}$.

3.2 Keyframes and Voxelization

Following prior work by James et al. [25], we construct a structured observation and action space through keyframe extraction and voxelization.

Training our agent to directly predict continuous actions is inefficient and noisy. So instead, for each demonstration ζ, we extract a set of keyframe actions $\{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_m\} \subset \mathcal{A}$ that capture bottlenecks [61] in the action sequence with a simple heuristic: an action is a keyframe if (1) the joint-velocities are near zero and (2) the gripper open state has not changed. Each datapoint in the demonstration ζ can then be cast as a "predict the next (best) keyframe action" task. See Appendix Figure F for an illustration of this process.

To learn action-centric representations [17] in 3D, we use a voxel grid [62, 63] to represent both the observation and action space. The observation voxels $\mathbf{v}$ are reconstructed from RGB-D observations $\tilde{o}$ fused through triangulation $\tilde{o} \Rightarrow \mathbf{v}$ from known camera extrinsics and intrinsics. By default, we use a voxel grid of 100^3, which corresponds to a volume of 1.0m^3 in metric scale. The keyframe actions $\mathbf{k}$ are discretized such that training our BC agent can be formulated as a "next best action" classification task. Translation is simply the closest voxel to the center of the gripper fingers. Rotation is discretized into 5 degree bins for each of the three rotation axes. Gripper open state is a binary value. Collide is also a binary value that indicates if the motion-planner should avoid everything in the voxel grid or nothing at all; switching between these two modes of collision avoidance is crucial as tasks often involve both contact based (e.g., pulling the drawer open) and non-contact based motions (e.g., reaching the handle without colliding into anything).

3.3 PERACT Agent

PERACT is a Transformer-based [2] agent that takes in a voxel observation and language goal $(\mathbf{v}, \mathbf{l})$, and outputs a discretized translation, rotation, and gripper open action. This action is executed with a motion-planner, after which this process is repeated until the goal is reached.

The language goal $\mathbf{l}$ is encoded with a pre-trained language model. We use CLIP's [64] language encoder, but any pre-trained language model would suffice [13, 59]. Our choice of CLIP opens up possibilities for future work to use pre-trained vision features that are aligned with the language for better generalization to unseen semantic categories and instances [16].

The voxel observation $\mathbf{v}$ is split into 3D patches of size 5^3 (akin to vision-transformers like ViT [4]). In implementation, these patches are extracted with a 3D convolution layer with a kernel-size and stride of 5, and further encoded with another 3D convolution layer. The language encodings are fine-tuned with a linear layer and then appended with the voxel encodings to form the input sequence. We also add learned positional embeddings to the sequence to incorporate voxel and token positions.

The input sequence of language and voxel encodings is extremely long. A standard Transformer with $\mathcal{O}(n^2)$ self-attention connections and an input of $(100/5)^3 = 8000$ patches is hard to fit on the memory of a commodity GPU. Instead, we use the Perceiver [1] Transformer. Perceiver is a latent-space Transformer, where instead of attending to the entire input, it first computes cross-attention between the input and a much smaller set of latent vectors (which are randomly initialized and trained). These latents are encoded with self-attention layers, and for the final output, the latents are again cross-attended with the input to match the input-size. See Appendix Figure 6 for an illustration. By default, we use 2048 latents of dimension 512 : $\mathbb{R}^{2048\times512}$, but in Appendix G we experiment with different latent sizes.

The Perceiver Transformer uses 6 self-attention layers to encode the latents and outputs a sequence of patch encodings. These patch encodings are upsampled with a 3D convolution layer and tri-linear upsampling. The decoder includes a skip-connection from the encoder (like in UNets [65]). The per-voxel encodings are then used to predict discretized actions. For translation, the voxel encodings are reshaped into the original voxel grid (100^3) to form a 3D $\mathcal{Q}$-function of action-values. For rotation, gripper open, and collide, the encodings are max-pooled and then decoded with linear layers to form their respective $\mathcal{Q}$-function. The best action $\mathcal{T}$ is chosen by simply maximizing the $\mathcal{Q}$-functions:

$$\mathcal{T}_{\text{trans}} = \underset{(x,y,z)}{\text{argmax}}\ \mathcal{Q}_{\text{trans}}((x,y,z) \mid \mathbf{v}, \mathbf{l}), \qquad \mathcal{T}_{\text{rot}} = \underset{(\psi,\theta,\phi)}{\text{argmax}}\ \mathcal{Q}_{\text{rot}}((\psi,\theta,\phi) \mid \mathbf{v}, \mathbf{l}),$$
$$\mathcal{T}_{\text{open}} = \underset{\omega}{\text{argmax}}\ \mathcal{Q}_{\text{open}}(\omega \mid \mathbf{v}, \mathbf{l}), \qquad \mathcal{T}_{\text{collide}} = \underset{\kappa}{\text{argmax}}\ \mathcal{Q}_{\text{collide}}(\kappa \mid \mathbf{v}, \mathbf{l}),$$

where (x, y, z) is the voxel location in the grid, (ψ, θ, ϕ) are discrete rotations in Euler angles, ω is the gripper open state and κ is the collide variable. See Figure 5 for examples of $\mathcal{Q}$-predictions.

3.4 Training Details

PERACT is trained through supervised learning with discrete-time input-action tuples from a dataset of demonstrations. These tuples are composed of voxel observations, language goals, and keyframe actions $\{(\mathbf{v}_1, \mathbf{l}_1, \mathbf{k}_1), (\mathbf{v}_2, \mathbf{l}_2, \mathbf{k}_2), \ldots\}$. During training, we randomly sample a tuple and supervise the agent to predict the keyframe action $\mathbf{k}$ given the observation and goal $(\mathbf{v}, \mathbf{l})$. For translations, the ground-truth action is represented as a one-hot voxel encoding $Y_{\text{trans}} : \mathbb{R}^{H\times W\times D}$. Rotations are also represented with a one-hot encoding per rotation axis with R rotation bins $Y_{\text{rot}} : \mathbb{R}^{(360/R)\times 3}$ ($R = 5$ degrees for all experiments). Similarly, open and collide variables are binary one-hot vectors $Y_{\text{open}} : \mathbb{R}^2$, $Y_{\text{collide}} : \mathbb{R}^2$. The agent is trained with cross-entropy loss like a classifier:

$$\mathcal{L}_{\text{total}} = -\mathbb{E}_{Y_{\text{trans}}}[\log\mathcal{V}_{\text{trans}}] - \mathbb{E}_{Y_{\text{rot}}}[\log\mathcal{V}_{\text{rot}}] - \mathbb{E}_{Y_{\text{open}}}[\log\mathcal{V}_{\text{open}}] - \mathbb{E}_{Y_{\text{collide}}}[\log\mathcal{V}_{\text{collide}}],$$

where $\mathcal{V}_{\text{trans}} = \text{softmax}(\mathcal{Q}_{\text{trans}}((x,y,z)|\mathbf{v},\mathbf{l}))$, $\mathcal{V}_{\text{rot}} = \text{softmax}(\mathcal{Q}_{\text{rot}}((\psi,\theta,\phi)|\mathbf{v},\mathbf{l}))$, $\mathcal{V}_{\text{open}} = \text{softmax}(\mathcal{Q}_{\text{open}}(\omega|\mathbf{v},\mathbf{l}))$, $\mathcal{V}_{\text{collide}} = \text{softmax}(\mathcal{Q}_{\text{collide}}(\kappa|\mathbf{v},\mathbf{l}))$ respectively. For robustness, we also augment $\mathbf{v}$ and $\mathbf{k}$ with translation and rotation perturbations. See Appendix E for more details.

By default, we use a voxel grid size of 100^3. We conducted validation tests by replaying expert demonstrations with discretized actions to ensure that 100^3 is a sufficient resolution for execution. The agent was trained with a batch-size of 16 on 8 NVIDIA V100 GPUs for 16 days (600K iterations). We use the LAMB [66] optimizer following Perceiver [1].

For multi-task training, we simply sample input-action tuples from all tasks in the dataset. To ensure that tasks with longer horizons are not over-represented during sampling, each batch contains a uniform distribution of tasks. That is, we first uniformly sample a set of tasks of batch-size length, then pick a random input-action tuple for each of the sampled tasks. With this strategy, longer-horizon tasks need more training steps for full coverage of input-action pairs, but all tasks are given equal weighting during gradient updates.

4 Results

We perform experiments to answer the following questions: (1) How effective is PERACT compared to unstructured image-to-action frameworks and standard architectures like 3D ConvNets? And what are the factors that affect PERACT's performance? (2) Is the global receptive field of Transformers actually beneficial over methods with local receptive fields? (3) Can PERACT be trained on real-world tasks with noisy data?

4.1 Simulation Setup

We conduct our primary experiments in simulation for the sake of reproducibility and benchmarking.

Environment. The simulation is set in CoppelaSim [67] and interfaced through PyRep [68]. All experiments use a Franka Panda robot with a parallel gripper. The input observations are captured from four RGB-D cameras positioned at the front, left shoulder, right shoulder, and on the wrist, as shown in Appendix Figure 8. All cameras are noiseless and have a resolution of 128×128.

Language-Conditioned Tasks. We train and evaluate on 18 RLBench [15] tasks. See Figure 1 for examples and Appendix A for details on individual tasks. Each task also includes several variations, ranging from 2-60 possibilities, e.g., in the `stack blocks` task, *"stack 2 red blocks"* and *"stack 4 purple blocks"* are two variants. These variants are randomly sampled during data generation, but kept consistent during evaluations for one-to-one comparisons. Some RLBench tasks were modified to include additional variations to stress-test multi-task and language-grounding capabilities. There are a total of 249 variations across 18 tasks, and the number of extracted keyframes range from 2-17 frames. Language goals are constructed from templates (but human-annotated for real-world tasks). Note that in all experiments, we do not test for generalization to unseen objects, i.e., our train and test objects are the same. However the poses, goals, and semantic instantiations (like color, shape, size, category) change across episodes. The focus here is to evaluate the effectiveness of a single multi-task agent trained on all tasks and variants.

Evaluation Metric. Each multi-task agent is evaluated independently on all 18 tasks. Evaluations are scored either 0 for failures or 100 for complete successes. There are no partial credits. We report average success rates on 25 evaluation episodes per task ($25 \times 18 = 450$ total episodes) for agents trained with $n = 10, 100$ demonstrations per task. During evaluation, an agent keeps taking actions until an oracle indicates task-completion or reaches a maximum of 25 steps.

4.2 Simulation Results

Table 1 reports success rates of multi-task agents trained on all 18 tasks. We could not investigate single-task agents due to resource constraints of training 18 individual agents.

Baseline Methods. We study the effectiveness of our problem formulation by benchmarking against two baselines: Image-BC and C2FARM-BC . Image-BC is an image-to-action agent similar to BC-Z [14]. Following BC-Z we use FiLM [69] for language conditioning, but the CNN vision encoders take in RGB-D images instead of just RGB. C2FARM-BC is a 3D fully-convolutional network by James et al. [25] that has achieved state-of-the-art results on RLBench tasks. Similar to our agent, C2FARM-BC also detects actions in a voxelized space, however it uses a coarse-to-fine-grain scheme to detect actions at two-levels of voxelization: 32^3 voxels with a 1^3m grid, and 32^3 voxels with a 0.15^3m grid after "zooming in" from the first level. Note that at the finest level C2FARM-BC has a higher resolution (0.47cm) than PERACT (1cm). We use the same ConvNet architecture as James et al. [25], but instead of training it with RL, we do BC with cross-entropy loss (from Section 3.4) and also condition it with language goals like in LingUNets [70, 16].

Multi-Task Performance. Table 1 compares the performance of Image-BC and C2FARM-BC against PERACT. With insufficient demonstrations, Image-BC has near zero performance on most tasks. Image-BC is disadvantaged with single-view observations and has to learn hand-eye coordination from scratch. In contrast, PERACT's voxel-based formulation naturally allows for in-

Method	open drawer		slide block		sweep to dustpan		meat off grill		turn tap		put in drawer		close jar		drag stick		stack blocks	
	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100
Image-BC	4	4	4	0	0	0	0	0	20	8	0	8	0	0	0	0	0	0
C2FARM-BC [25]	28	20	12	16	4	0	40	20	60	68	12	4	28	24	**72**	24	4	0
PERACT (w/o Lang)	20	28	8	12	20	16	40	48	36	60	16	16	16	12	48	60	0	0
PERACT	**68**	**80**	**32**	**72**	**72**	**56**	**68**	**84**	**72**	**80**	**16**	**68**	**32**	**60**	36	**68**	**12**	**36**

	screw bulb		put in safe		place wine		put in cupboard		sort shape		push buttons		insert peg		stack cups		place cups	
	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100
Image-BC	0	0	0	4	0	0	0	0	0	0	4	0	0	0	0	0	0	0
C2FARM-BC [25]	12	8	0	12	**36**	8	**4**	0	8	8	**88**	**72**	0	**4**	0	0	0	0
PERACT (w/o Lang)	0	**24**	8	20	8	**20**	0	0	0	0	60	68	4	0	0	0	0	0
PERACT	**28**	**24**	**16**	**44**	20	12	0	**16**	**16**	**20**	56	48	**4**	0	0	0	0	0

Table 2. **Multi-Task Test Results.** Success rates (mean %) of various multi-task agents tasks trained with either 10 or 100 demonstrations per task and evaluated on 25 episodes per task. Each evaluation episode is scored either a 0 for failure or 100 for complete succces. PERACT outperforms C2FARM-BC [25], the most competitive baseline, with an average improvement of $+168\%$ with 10 demonstrations and $+266\%$ with 100 demonstrations across all tasks.

tegrating multi-view observations, learning 6-DoF action representations, and data-augmentation in 3D, all of which are non-trivial to achieve in image-based methods. C2FARM-BC is the most competitive baseline, but it has a limited receptive field because of the coarse-to-fine-grain scheme and partly due to the convolution-only architecture. PERACT outperforms C2FARM-BC in $25/36$ evaluations in Table 1 with **an average improvement** of $\mathbf{+168\%}$ **with 10 demonstrations and** $\mathbf{+266\%}$ **with 100 demonstration**. For a number of tasks, C2FARM-BC actually performs worse with more demonstrations, likely due to insufficient capacity. Since additional training demonstrations include additional task variants to optimize for, they might end up hurting performance.

In general, 10 demonstrations are sufficient for PERACT to achieve $> 65\%$ success on tasks with limited variations like `open drawer` (3 variations). But tasks with more variations like `stack blocks` (60 variations) need substantially more data, sometimes to simply cover all possible concepts like "*teal color block*" that might have not appeared in the training data. For three tasks: `insert peg`, `stack cups`, and `place cups`, all agents achieve near zero success. These are very high-precision tasks where being off by a few centimeters or degrees could lead to unrecoverable failures. See the simulation rollouts in the supplementary video to get a sense of the complexity of these evaluation tasks.

Ablations. Table 1 reports PERACT w/o Lang, an agent without any language conditioning. Without a language goal, the agent does not know the underlying task and performs at chance. We also report additional ablation results on the `open drawer` task in Figure 3. To summarize these results: (1) the skip connection helps train the agent slightly faster, (2) the Perceiver Transformer is crucial for achieving good performance with the global receptive field, and (3) extracting good keypoint actions is essential for supervised training as randomly chosen or fixed-interval keypoints lead to zero-performance.

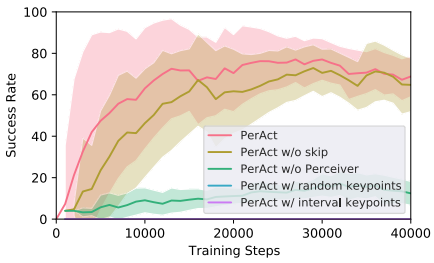

Figure 3. **Ablation Experiments.** Success rate of PERACT after ablating key components.

Sensitivity Analysis. In Appendix G we investigate factors that affect PERACT's performance: the number of Perceiver latents, voxelization resolution, and data augmentation. We find that more latent vectors generally improve the capacity of the agent to model more tasks, but for simple short-horizon tasks, fewer latents are sufficient. Similarly, with different voxelization resolutions, some tasks are solvable with coarse voxel grids like 32^3, but some high-precision tasks require the full 100^3 grid. Finally, rotation perturbations in the data augmentation generally help in improving robustness essentially by exposing the agent to more rotation variations of objects.

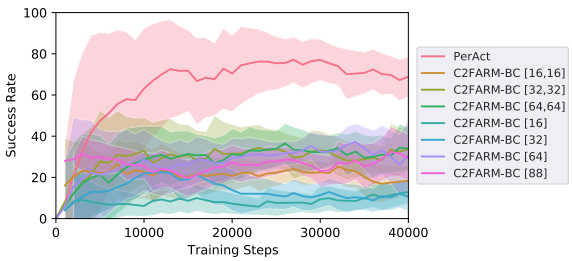

Figure 4. **Global vs. Local Receptive Field Experiments.** Success rates of PERACT against various C2FARM-BC [25] baselines

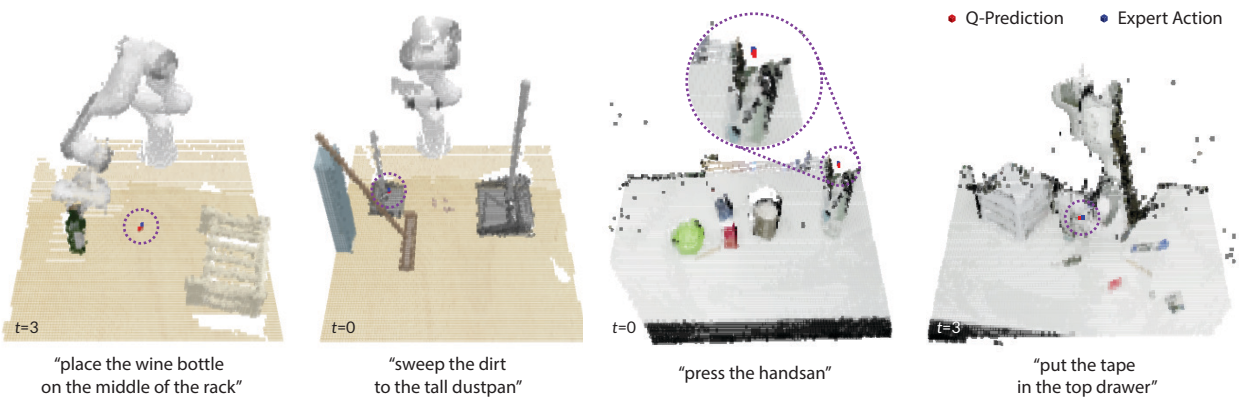

Figure 5. **Q-Prediction Examples**: Qualitative examples of translation $\mathcal{Q}$-Predictions from PERACT along with expert actions, highlighted with dotted-circles. The left two are simulated tasks, and the right two are real-world tasks. See Appendix H for more examples.

4.3 Global vs. Local Receptive Fields

To further investigate our Transformer agent's global receptive field, we conduct additional experiments on the `open drawer` task. The `open drawer` task has three variants: *"open the top drawer"*, *"open the middle drawer"*, and *"open the bottom drawer"*, and with a limited receptive field it is hard to distinguish the drawer handles, which are all visually identical. Figure 4 reports PERACT and C2FARM-BC agents trained with 100 demonstrations. Although the `open drawer` tasks can be solved with fewer demonstrations, here we want to ensure that insufficient data is not an issue. We include several versions of C2FARM-BC with different voxelization schemes. For instance, $[16, 16]$ indicates two levels of 16^3 voxel grids at 1m^3 and 0.15m^3, respectively. And $[64]$ indicates a single level of a 64^3 voxel grid without the coarse-to-fine-grain scheme. PERACT is the only agent that achieves $> 70\%$ success, whereas all C2FARM-BC versions perform at chance with $\sim 33\%$, indicating that the global receptive field of the Transformer is crucial for solving the task.

4.4 Real-Robot Results

We also validated our results with real-robot experiments on a Franka Emika Panda. See Appendix D for setup details. Without any sim-to-real transfer or pre-training, we trained a multi-task PERACT agent from scratch on 7 tasks with a total of just 53 demonstrations. See the supplementary video for qualitative results that showcase the diversity of tasks and robustness to scene changes. Table 3 reports success rates from small-scale evaluations. Similar to the simulation results, we find that PERACT is able to achieve $> 65\%$ success on simple short-horizon tasks like pressing hand-sanitizers from just a handful number of demonstrations. The most common failures involved predicting incorrect gripper open actions, which often lead the agent into unseen states. This could be addressed in future works by using HG-DAgger style approaches to correct the agent [14]. Other issues included the agent exploiting biases in the dataset like in prior work [16]. This could be addressed by scaling up expert data with more diverse tasks and task variants.

Task	# Train	# Test	Succ. %
Press Handsan	5	10	90
Put Marker	8	10	70
Place Food	8	10	60
Put in Drawer	8	10	40
Hit Ball	8	10	60
Stack Blocks	10	10	40
Sweep Beans	8	5	20

Table 3. Success rates (mean %) of a multi-task model trained an evaluated 7 real-world tasks (see Figure 1).

5 Conclusion and Limitations

We presented PERACT, a Transformer-based multi-task agent for 6-DoF manipulation. Our experiments with both simulated and real-world tasks indicate that the right problem formulation, i.e., detecting voxel actions, makes a substantial difference in terms of data efficiency and robustness.

While PERACT is quite capable, extending it to dexterous continuous control remains a challenge. PERACT is at the mercy of a sampling-based motion-planner to execute discretized actions, and is not easily extendable to N-DoF actuators like multi-fingered hands. See Appendix J for an extended discussion on PERACT's limitations. But overall, we are excited about scaling up robot learning with Transformers by focusing on *diverse* rather than narrow multi-task data for robotic manipulation.

References

[1] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2017.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Neural Information Processing Systems (NeurIPS)*, 2020.

[4] 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. In *International Conference on Learning Representations (ICLR)*, 2020.

[5] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

[6] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, 2, 2019.

[7] T. Chen, S. Saxena, L. Li, D. J. Fleet, and G. Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021.

[8] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *In Neural Information Processing Systems (NeurIPS)*, 2021.

[9] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[10] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[11] 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 (ICML)*, 2021.

[12] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2020.

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

[14] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning (CoRL)*, 2021.

[15] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[16] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *In Conference on Robot Learning (CoRL)*, 2021.

[17] J. J. Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[20] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[21] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *2017 IEEE international conference on robotics and automation (ICRA)*, 2017.

[22] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[23] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, 2020.

[24] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *Conference on Robot Learning (CoRL)*, 2020.

[25] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[26] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *Conference on Robot Learning (CoRL)*, 2018.

[27] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. In *Robotics: Science and Systems (RSS)*, 2020.

[28] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[29] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[30] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[31] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox. 6-dof grasping for target-driven object manipulation in clutter. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[32] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *International Conference on Computer Vision (ICCV)*, 2019.

[33] Z. Xu, H. Zhanpeng, and S. Song. Umpnet: Universal manipulation policy network for articulated objects. *Robotics and Automation Letters (RA-L)*, 2022.

[34] 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. *arXiv preprint arXiv:2112.05124*, 2021.

[35] 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.

[36] W. Yuan, C. Paxton, K. Desingh, and D. Fox. Sornet: Spatial object-centric representations for sequential manipulation. In *In Conference on Robot Learning (CoRL)*. PMLR, 2021.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[38] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *In International Conference on Computer Vision (ICCV)*, 2021.

[39] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Neural information processing systems (NeurIPS)*, 2021.

[40] K.-H. Lee, O. Nachum, M. Yang, L. Lee, D. Freeman, W. Xu, S. Guadarrama, I. Fischer, E. Jang, H. Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

[41] H. M. Clever, A. Handa, H. Mazhar, K. Parker, O. Shapira, Q. Wan, Y. Narang, I. Akinola, M. Cakmak, and D. Fox. Assistive tele-op: Leveraging transformers to collect robotic task demonstrations. *arXiv preprint arXiv:2112.05129*, 2021.

[42] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. *arXiv preprint arXiv:2107.03996*, 2021.

[43] D. S. Chaplot, D. Pathak, and J. Malik. Differentiable spatial planning using transformers. In *International Conference on Machine Learning (ICML)*, 2021.

[44] J. J. Johnson, L. Li, A. H. Qureshi, and M. C. Yip. Motion planning transformers: One model to plan them all. *arXiv preprint arXiv:2106.02791*, 2021.

[45] S. Dasari and A. Gupta. Transformers for one-shot visual imitation. *arXiv preprint arXiv:2011.05970*, 2020.

[46] H. Kim, Y. Ohmura, and Y. Kuniyoshi. Transformer-based deep imitation learning for dual-arm robot manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[47] Y. Han, R. Batra, N. Boyd, T. Zhao, Y. She, S. Hutchinson, and Y. Zhao. Learning generalizable vision-tactile robotic grasping strategy for deformable objects via transformer. *arXiv preprint arXiv:2112.06374*, 2021.

[48] M. Shridhar and D. Hsu. Interactive visual grounding of referring expressions for human-robot interaction. In *Robotics: Science and Systems (RSS)*, 2018.

[49] C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[50] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pages 481–495. Springer, 2013.

[51] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research (IJRR)*, 2016.

[52] Y. Bisk, D. Yuret, and D. Marcu. Natural language communication with robots. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2016.

[53] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone. Learning to interpret natural language commands through human-robot dialog. In *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[54] J. Hatori, Y. Kikuchi, S. Kobayashi, K. Takahashi, Y. Tsuboi, Y. Unno, W. Ko, and J. Tan. Interactively picking real-world objects with unconstrained spoken language instructions. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[55] Y. Chen, R. Xu, Y. Lin, and P. A. Vela. A Joint Network for Grasp Detection Conditioned on Natural Language Commands. *arXiv:2104.00492 [cs]*, Apr. 2021.

[56] V. Blukis, R. A. Knepper, and Y. Artzi. Few-shot object grounding for mapping natural language instructions to robot control. In *Conference on Robot Learning (CoRL)*, 2020.

[57] C. Paxton, Y. Bisk, J. Thomason, A. Byravan, and D. Fox. Prospection: Interpretable plans from language by predicting the future. In *International Conference on Robotics and Automation (ICRA)*, 2019.

[58] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[59] C. Lynch and P. Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[60] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *arXiv preprint arXiv:2112.03227*, 2021.

[61] E. Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *International Conference on Robotics and Automation (ICRA)*, 2021.

[62] H. Moravec. Robot spatial perceptionby stereoscopic vision and 3d evidence grids. *Perception*, 1996.

[63] Y. Roth-Tabak and R. Jain. Building an environment model using depth information. *Computer*, 22(6):85–90, 1989.

[64] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020*, 2021.

[65] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[66] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

[67] E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[68] S. James, M. Freese, and A. J. Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019.

[69] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

[70] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In *2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[71] S. James and P. Abbeel. Coarse-to-fine q-attention with learned path ranking. *arXiv preprint arXiv:2204.01571*, 2022.

[72] S. James and P. Abbeel. Bingham policy parameterization for 3d rotations in reinforcement learning. *arXiv preprint arXiv:2202.03957*, 2022.