PDFactor: Learning Tri-Perspective View Policy Diffusion Field for Multi-Task Robotic Manipulation

Supplementary Material

A. Task Details

A.1. RLBench Tasks

In this section, we provide a concise overview of the RLBench [31] dataset. Tab. 7 is an overview of the 18 selected tasks we use in the experiments. Task variations include randomly sampled colors, sizes, shapes, counts, placements, and categories of objects. The color set comprises 20 options: colors = {`red`, `maroon`, `lime`, `green`, `blue`, `navy`, `yellow`, `cyan`, `magenta`, `silver`, `gray`, `orange`, `olive`, `purple`, `teal`, `azure`, `violet`, `rose`, `black`, `white`}. The size set includes two options: sizes = {`short`, `tall`}, while the shape set contains five: shapes = {`cube`, `cylinder`, `triangle`, `star`, `moon`}. Object counts can range between: counts = {1, 2, 3}. Placement and category details are task-specific, and objects are positioned on the tabletop in random poses. However, for larger objects such as drawers, pose variations are constrained to ensure kinematic feasibility when manipulated by the Franka robotic arm.

In the ablation study, we adopt task classification from [25, 41] to categorize 18 RLBench tasks listed in Tab. 7 into 8 groups based on their primary challenges. These task groups consist of

- The **Planning** group contains tasks with multiple subgoals. The included tasks are: `meat off grill` and `push buttons`.
- The **Tools** group is a special case of planning where a robot must grasp an object to interact with the target object. The included tasks are: `slide block`, `drag stick` and `sweep to dustpan`.
- The **Long term** group requires more than 10 macro-steps to be completed. The included tasks are: `put in drawer`, `place cups` and `stack blocks`.
- The **Rotation-invariant** group can be solved without changes in the gripper rotation. The included tasks are: `push buttons`.
- The **Motion planner** group requires precise grasping. The included tasks are: `open drawer`, `place wine`, `close jar` and `put item in drawer`.
- The **Multimodal** group can have multiple possible trajectories to solve a task due to a large affordance area of the target object. The included tasks are: `turn tap` and `stack cups`.
- The **Precision** group involves precise object manipulation. The included tasks are: `insert peg` and `sort shape`.
- The **Visual Occlusion** group involves tasks with large objects and thus there are occlusions from certain views. The included tasks are: `put in safe` and `put in cupboard`.

A.2. Real-World Tasks

Tab. 8 summarize the tasks used for the real-world evaluation. For data collection, we randomly sample a task variation and place objects on the table in a random configuration. A sequence of end-effector poses is collected by kinesthetically moving the robot arm around. Then the robot is reset to start pose and controlled to sequentially move to each target pose, while the RGB-D stream from the camera is simultaneously recorded.

B. Additional Details on Baselines

We provide more details for the baselines used in this paper:

- **C2F-ARM-BC** [32] iteratively voxelizes RGB-D images and predicts actions in a coarse-to-fine manner. Q-values are estimated within each voxel and the translation action is determined by the centroid of the voxel with the maximal Q-values.
- **PerAct** [58] voxelizes the workspace and utilize Perceiver Transformer to predict a 3D action value map.
- **Act3D** [19] featurizes the robot's 3D workspace using coarse-to-fine point cloud sampling and featurization.
- **3D Diffuser Actor** [36] leverages a diffusion transformer to iteratively transform pure noise to real action by combining 3D point cloud representation with diffusion objective.
- **RVT** [21] projects calibrated point cloud to orthogonal views via point splatting and deploys a multi-view transformer to predict actions heatmaps which are then fused across views by back-projecting to 3D.
- **RVT-2** [22] adopts a two-stage prediction strategy by zooming in the point cloud centered at the action predicted from the first stage. Each stage shares the same architecture with RVT.

C. Additional Implementation Details

C.1. Training Pipeline

To simplify the tasks, keyframe-based policies assume access to a motion planner, so that a demonstrations can be split into several observation-action pairs (*i.e.*, keyframes). Keyframes are identified based on empirical rules: an action is a keyframe if (1) the joint-velocities are near zero and (2) the gripper open state has not changed. To learn the

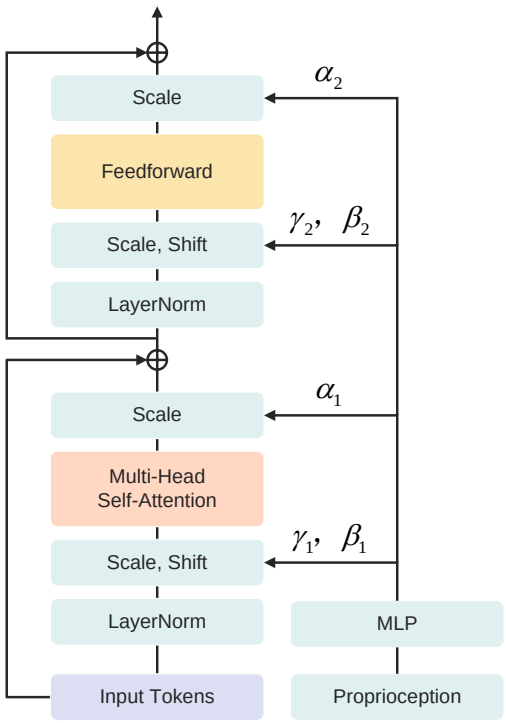

Figure 6. **Transformer Block.**

policy from demonstrations, we uniformly sample a group of expert episodes from all the task variations, and then random choose a keyframe or a non-keyframe with a predefined probability for each episode to form a batch. A non-keyframe is an observation at any time paired with the nearest next keyframe action.

C.2. Model Details

Preprocessing. The observed RGB-D images are first encoded with two convolution layer and a SiLU activation to upsample the channel dimension from 6 to 96. The 6 channels are composed of: 3 RGB and 3 point coordinates. The point coordinates are Cartesian coordinates in the robot's coordinate frame. The triplane features are constructed by projecting and max-pooling calibrated point cloud features to the corresponding triplane grid. The projected triplane features are split into patches through a convolution layer with a kernel size same as stride. The language goals are encoded with CLIP's language encoder. The robot proprioception, which includes 4 scalar values: gripper open, left finger joint position, right finger joint position, and timestep, is encoded by a MLP consisting of 2 linear layer and a SiLU activation.

Architecture. For the tri-perspective view transformer, we adopt standard transformer architecture with adaptive layer norm. Fig. 6 shows the details of the transformer block we used in PDFactor model. Notably, to avoid gradient instability or numerical overflow, we apply QKNorm [28] when calculating attention. For the denoising MLP, the noisy action is first passed through a linear layer and then several MLP blocks. Each block sequentially applies an adaptive layer norm layer, a linear layer, a SiLU activation and another linear layer, merging with a residual connection. The latent vector $\mathbf{z}$ is added to the timestep embedding, which serves as condition of the denoising MLP via adaptive layer norm. Finally, a linear layer is applied to predict noise component and binary states.

Hyperparameter	Value
Model	
image resolution	128×128
triplane resolution (H, W, D)	128,128,128
patch size	4
MLP ratio	4.0
diffusion timesteps	100
noise scheduler (position)	scaled linear
noise scheduler (rotation)	cosine
rotation representation	6D
Training	
keyframe sampling ratio	0.8
training iteration	30k
batch size	256
optimizer	8bit AdamW
learning rate	2.5×10^{-4}
weight decay	1.0×10^{-4}
momentum	(0.9, 0.95)
EMA	0.9999
λ_1	100
λ_2,λ_3	1

Table 6. **Hyperparameters.**

C.3. Hyperparameters

The hyperparameters used in PDFactor are shown in Tab. 6. Other hyperparameters are in line with previous works [21, 22, 58] for fair comparison.

D. Additional Qualitative Analysis

We provide 7 representative task episodes generated by our PDFactor in RLBench simulation and 6 real-world task demo in the attached videos (demo_sim.mp4 and demo_real.mp4). In both simulation and real-world scenarios, our PDfactor succeeds in solving tasks that require long-term understanding, *e.g.*, `stack blocks`, `stack cups` and `place cups`. Besides, in the tasks that require high precision, *e.g.*, `insert peg`, `sort shape` and `place cups`, our model achieves preferable performance.

Task	Variation Type	# Variations	Avg. Keyframes	Language Template
open drawer	placement	3	3.0	“open the __ drawer”
slide block	color	4	4.7	“slide the block to __ target”
sweep to dustpan	size	2	4.6	“sweep dirt to the __ dustpan”
meat off grill	category	2	5.0	“take the __ off the grill”
turn tap	placement	2	2.0	“turn __ tap”
put in drawer	placement	3	12.0	“put the item in the __ drawer”
close jar	color	20	6.0	“close the __ jar”
drag stick	color	20	6.0	“use the stick to drag the cube onto the __ target”
stack blocks	color,count	60	14.6	“stack __ __ blocks”
screw bulb	color	20	7.0	“screw in the __ light bulb”
put in safe	placement	3	5.0	“put the money away in the safe on the __ shelf”
place wine	placement	3	5.0	“stack the wine bottle to the __ of the rack”
put in cupboard	category	9	5.0	“put the __ in the cupboard”
sort shape	shape	5	5.0	“put the __ in the shape sorter”
push buttons	color	50	3.8	“push the __ button, [then the __ button]”
insert peg	color	20	5.0	“put the ring on the __ spoke”
stack cups	color	20	10.0	“stack the other cups on top of the __ cup”
place cups	count	3	11.5	“place __ cups on the cup holder”

Table 7. **RLBench task details.**

Task	Variation Type	# Variations	Avg. Keyframes	Language Template
put fruit	category	4	7.7	“put __ on plate”
push buttons	color	5	9.4	“push the __ button, [then the __ button]”
stack cups	color	5	15.2	“stack __ cups on __ cup”
stack blocks	color	5	14.0	“stack __ blocks on green cylinder”
sort cylinder	placement	1	8.4	“put cylinder in shape sorter”
put mustard	placement	1	9.1	“put mustard on shelf”

Table 8. **Real-world task details.**