SD^{2}Actor: Continuous State Decomposition via Diffusion Embeddings for Robotic Manipulation

Supplementary Material

A. Task Descriptions

A.1. Simulation Tasks

We follow the eight tasks with mulitple goal states in ARNOLD. Specifically, we focus on continuous goal states and define success ranges around them wherein robots should maintain object states for 2 seconds to succeed. Table 9 provides an overview. In these templates, "[value_object]" holds for the actual object name, *e.g.*, "white bottle". The goal states are specified by "[value_height]", "[value_degree]", and "[value_percent]". The placeholders are lexicalized randomly by sampling from candidate pools which contain equivalent phrases, *e.g.*, "fifty percent", "half ", "two quarters". More task details follows:

- In PICKUP OBJECT and REORIENT OBJECT, we instruct the robot to manipulate a bottle to achieve different goals.
- In the four tasks {OPEN,CLOSE}{DRAWER,CABINET}, the goal value specifies the geometric state of the articulated joint, either in terms of distance (for prismatic joints in DRAWER) or angle (for revolute joints in CABINET).
- In POUR WATER and TRANSFER WATER, the manipulated object is a cup containing water, and the goal specifies the percentage of water to be poured out (POUR) or poured into another cup (TRANSFER).

The motion planner for these tasks consists of four or seven sub-task stages. The each stage is beginning with the current visual observation and ending with a consequent end effector pose. In practical, we extract two observation-action pairs o_1, a_1, o_2, a_2 for two-phase learning, which denote the pre-grasp phase of reaching the object and the phase of grasping the object to reach the target state. The execution of a_1 and a_2 is scheduled by the motion planner.

A.2. Real Robot Tasks

In order to be consistent with the simulated experiments, we have set a similar goal state on the real-world tasks and defined a success range for the execution results. As shown in the Table 8, as this article focuses on modeling object states, we set the shape of the objects in the data to be similar for the same task. The success ranges for each task are consistent with those in the simulation.

- In Pickup object, Translate object and Reorient object, we guide the robot to manipulate(grasp) the bottle to translate to a specified position or rotate to a certain orientation, with each of its positions corresponding to a different state.
- In Slide block, the robot mainly relies on the closed grippers to push the block to move. The goal value specifies the distance the block will move.
- In Rotate pen, the manipulated object is a pen that grasps the tail end and rotates it only in the same plane, and its initial state is set to zero.

These five tasks have different levels of rotation and translation, and our model is trained on these data and tested for generalizability on continuous goal states. The results can indicate that our model can effectively capture rotational and translational goal states, and has some state generalization ability on robot manipulation tasks.

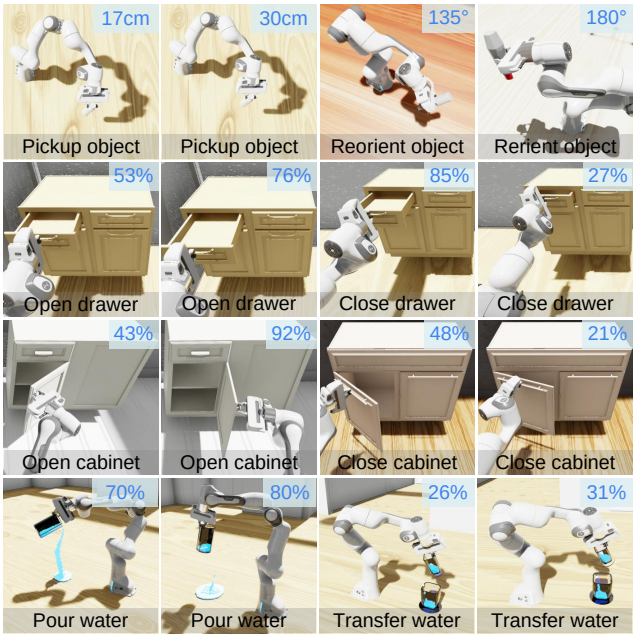

Figure 8. **Qualitative result of our SD^{2}Actor on any state splits.** We visualized 8 tasks which have continuous states. Blue figures are the state value.

Task Types	Instruction template
PickupObject	*Raise the bottle* [value_height] *above the table*
ReorientObject	*Reorient the bottle* [value_degree] *away from the up axis*
TranslateObject	*Translate the bottle* [value_length]
SlideBlock	*Slide the block* [value_percent] *from the current position*
RotatePen	*Rotate the pen* [value_percent] *from the current position*

Table 8. **Overview of the 5 tasks in Real Robot.**

Task Types	Goal States	Success Ranges	Instruction template
PickupObject	10, 20, 30, 40 (cm)	±5cm	*Raise* [value_object] [value_height] *above the ground*
ReorientObject	0, 45, 135, 180 (°)	±20°	*Reorient* [value_object] [value_degree] *away from the up axis*
OpenDrawer	25, 50, 75, 100 (%)	±10%	*Open the* [value_position] [value_object] [value_percent]
CloseDrawer	0, 25, 50, 75 (%)	±10%	*Close the* [value_position] [value_object] [value_percent]
OpenCabinet	25, 50, 75, 100 (%)	±10%	*Open the* [value_position] [value_object] [value_percent]
CloseCabinet	0, 25, 50, 75 (%)	±10%	*Close the* [value_position] [value_object] [value_percent]
PourWater	25, 50, 75, 100 (%)	±10%	*Pour* [value_percent] *water out of* [value_object]
TransferWater	20, 40, 60, 80 (%)	±10%	*Transfer* [value_percent] *water to* [value_object]

Table 9. **Overview of the 8 tasks in ARNOLD.** Each task features 4 goal states specified by human language, one of which is reserved for novel state evaluation. The task is deemed successful when the object state remains in the success range for two seconds. A few examples of delexicalized instruction templates for various tasks.

B. Implementation Detail

B.1. Prompts on LLMs

When the model can already generate accurate keyposes in a limited number of base states, we want to generalize to new states. Therefore, we utilize LLMs to extract novel states from the language and represent them as observed states. This requires a series of prompts to bootstrap the LLMs. Here are the specific Close Cabinet prompts:

```
I will give you an example included in
    ####/####.
###
Now you are given four instructions.
1.'adjust the closet 25% open.'
2.'adjust the cupboard 50% open.'
3.'adjust the cabinet 100% opened.'
Note that open and close are opposite
    quantities. For example, 25% open
    corresponds to 75% close. Don't get
    confused.
So after analyzing these three
    instructions, the states
    corresponding to the objects goal
    states are 0.25, 0.5 and 1
    respectively.
Now there is a new instruction 'open
    the cabinet 91%'. It can be deduced
    that the goal of the new instruction
     is the state
where the object is open to 0.91, and
    the closest states to the above are
    open 1 and 0.5, so the 0.91 state
    can be decomposed into 0.18 states
    0.5 and 0.82 states 1,
i.e., 0.91=0.18*0.5+0.82*1. Then the
    return value is 0.18*0.5+0.82*1. The
     return values are then determined
    by the two most relevant The return
    value consists of the indexes i, j
of the two most relevant instructions
    and the corresponding state
    decomposition coefficients a, b in
    the format <<<(i:a;j:b)>>>.
For example, the return value in this
    case is <<<(2:0.18;3:0.82)>>>.
###
If the new instruction is {instr}
    combined with the above four
    instructions then what is the return
     value? Calculate carefully, please.
You only need to return the right
    results!
```

We input all new state instructions into ChatGPT to decompose the state, take out the corresponding embeddings based on the decomposition coefficients and indexes and perform affine combination to generalize to the new state.

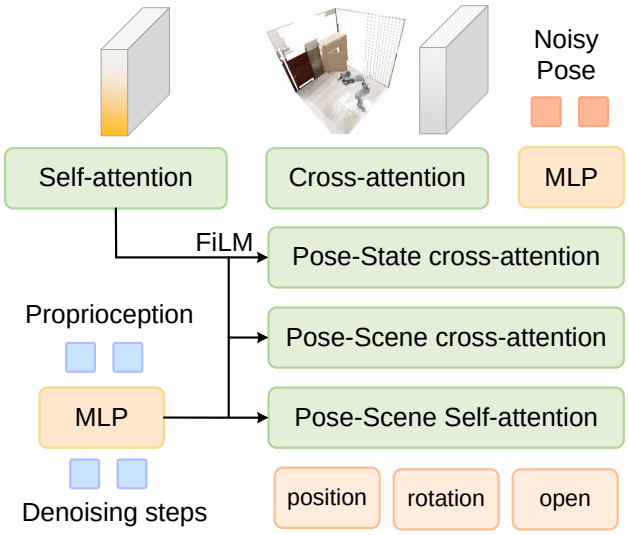

Figure 9. **The detailed architecture of our approach.**

	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	Average
6D-CLIPort[54]	6.72	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.84
PerAct[33]	94.78	24.68	36.13	60.14	23.19	30.10	49.25	28.57	43.36
PerAct(MT)[33]	90.30	14.29	25.21	33.78	20.29	19.42	26.87	17.86	31.00
3D Diffuser Actor[20]	89.55±1.67	19.48±0.55	35.29±2.12	61.49±0.62	20.29±2.78	27.18±2.82	52.24±0.67	32.14±0.96	42.21
SD^{2}Actor	**95.52**±0.19	**25.97**±2.24	**50.42**±0.24	**70.27**±0.61	**31.88**±0.70	**36.89**±0.00	**55.22**±2.98	**35.71**±0.24	**50.23**
SD^{2}Actor(MT)	91.79±1.67	15.58±2.25	38.66±0.00	45.27±0.00	27.54±0.67	25.24±0.00	29.85±2.98	16.07±0.17	36.25

	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	Average
6D-CLIPort[54]	25.37	0.00	0.00	2.70	0.00	5.83	0.00	7.14	5.13
PerAct[33]	95.52	28.57	52.94	68.24	49.28	48.54	85.07	53.57	60.22
PerAct(MT)[33]	92.54	20.78	47.90	56.76	39.13	37.86	64.18	30.36	48.69
3D Diffuser Actor[20]	91.79±0.73	20.78±0.00	56.30±0.24	71.62±0.61	42.03±0.70	44.66±0.07	86.57±0.00	58.93±1.69	59.08
SD^{2}Actor	**97.76**±0.74	**29.87**±2.25	**73.95**±0.24	**81.76**±1.37	**60.87**±0.63	**55.34**±1.25	**92.54**±2.98	**66.07**±0.00	**69.77**
SD^{2}Actor(MT)	97.01±2.36	24.68±0.00	67.23±0.24	68.92±0.00	52.17±2.78	43.69±0.31	67.16±0.73	32.14±1.69	56.62

Table 10. **The detailed base tasks performance.** Each task includes mean and variance.

	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	Average
6D-CLIPort[54]	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.09
PerAct[33]	0.68	0.48	10.06	13.58	0.00	0.00	2.15	5.88	4.10
PerAct(MT)[33]	2.04	0.95	9.20	6.98	0.00	2.78	6.45	1.68	3.76
3D Diffuser Actor[20]	1.02±0.73	1.43±1.07	10.34±0.94	13.96±2.44	1.24±1.40	1.39±0.00	2.69±0.00	6.72±0.00	4.85
SD^{2}Actor	**3.40**±0.04	**10.95**±0.67	**25.86**±0.03	**26.04**±0.01	**8.71**±0.21	**5.56**±0.13	**7.53**±0.10	**12.61**±0.95	**12.58**
SD^{2}Actor(MT)	1.02±0.15	7.14±0.07	16.67±0.03	20.75±0.01	6.64±0.06	1.39±5.80	1.61±0.86	8.40±0.24	7.95

	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	Average
6D-CLIPort[54]	0.00	0.00	0.57	1.13	0.83	2.78	0.00	16.81	2.77
PerAct[33]	2.38	0.00	12.93	18.11	6.22	8.33	1.61	2.52	6.51
PerAct(MT)[33]	3.06	2.38	18.68	11.13	3.73	11.11	9.14	4.20	7.93
3D Diffuser Actor[20]	1.70±0.04	0.48±0.68	13.79±0.25	18.30±0.19	7.47±0.06	12.50±0.00	1.61±0.09	4.20±0.94	7.51
SD^{2}Actor	**5.44**±0.00	**16.19**±1.20	**32.47**±0.11	**35.85**±0.01	**17.01**±0.22	**16.67**±0.00	**9.68**±0.87	**17.65**±1.88	**18.87**
SD^{2}Actor(MT)	2.38±0.15	13.81±0.08	29.31±0.25	26.60±0.05	12.45±0.24	9.72±2.58	1.61±0.37	15.13±0.00	13.88

Table 11. **The detailed novel states performance.** Each task includes mean and variance.

Hyper-parameter	Value
n	3
Batch size	192
Traing iteration	80K
Optimizer	AdamW
Learning rate	$1\times e^{-4}$
Weight Decay	$1\times e^{-5}$
Translation loss coefficient	30
Rotation loss coefficient	1
Openness loss coefficient	1
Rotation loss coefficient	1
Inter-task orthogonalization loss coefficient	5
Intra-task orthogonalization loss coefficient	1
Denoising steps	100
Image size	128
Embedding dim	512
GPU	RTX 3090

Table 12. **Hyper-parameters of SD^{2}Actor.**

B.2. Detailed Model Diagram

We present a more detailed architecture diagram of our State-conditioned Diffusion policy in Figure 9. The inputs to our diffusion are visual tokens, language tokens, states tokens, agent proprioception, the current noisy estimates of position and rotation and the denoising step. The visual tokens cross-attend to the language tokens and get residually updated. The proprioception tokens attend to the visual tokens to contextualize with the scene information. We subsample a number of visual tokens using Farthest Point Sampling. The sampled visual tokens, proprioception tokens and noisy pose tokens attend to each other. We modulate the attention using adaptive layer normalization and FiLM. Lastly, the contextualized noisy estimates are fed to MLP to predict the error terms as well as the openness.

B.3. Hyper-parameters for experiments

To enhance the reproducibility of SD^{2}Actor, we summarize the experiment details such as the configurations and parameters used in Table 12.

	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	Average
6D-CLIPort[54]	10.45	1.30	0.84	0.68	0.00	0.00	0.00	0.00	1.66
PerAct[33]	48.51	14.29	21.01	23.65	4.35	6.80	26.87	14.29	19.97
PerAct(MT)[33]	46.27	12.99	12.61	14.86	4.35	4.85	16.42	3.57	14.49
3D Diffuser Actor[20]	47.01±2.86	15.58±0.00	22.69±3.87	25.00±1.37	8.70±2.82	8.74±4.12	28.36±8.76	16.07±7.92	21.52
SD^{2}Actor	**50.75**±2.98	**19.48**±3.21	**36.97**±0.95	**37.84**±1.37	**14.49**±6.21	**12.62**±0.31	**31.34**±0.75	**19.64**±9.54	**27.89**
SD^{2}Actor(MT)	49.25±1.29	16.88±0.56	29.41±0.24	29.05±1.39	13.04±2.80	9.71±5.02	25.37±6.72	16.07±9.31	23.60

	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	Average
6D-CLIPort[54]	29.10	2.60	0.00	1.35	5.80	2.91	1.49	7.14	6.30
PerAct[33]	47.76	14.29	33.61	28.38	24.64	13.59	31.34	19.64	26.66
PerAct(MT)[33]	47.01	12.99	23.53	26.35	5.80	8.74	25.37	5.36	19.39
3D Diffuser Actor[20]	47.76±0.18	16.88±1.94	35.29±0.24	31.08±5.49	21.74±2.52	15.53±3.31	34.33±5.49	21.43±7.99	28.01
SD^{2}Actor	**51.49**±0.73	**24.68**±5.07	**52.94**±0.00	**50.00**±1.04	**36.23**±3.01	**23.30**±2.82	**38.81**±2.96	**30.36**±6.57	**38.48**
SD^{2}Actor(MT)	49.25±0.64	20.78±2.56	46.22±2.88	46.62±0.61	28.99±1.71	18.45±2.84	28.36±2.96	25.00±0.00	32.96

Table 13. **The detailed any states performance.** Each task includes mean and variance.

Pickup object	Reorient object	Open drawer	Close drawer	Open cabinet	Close cabinet	Pour water	Transfer water
± 5cm	± 20°	± 10%	± 10%	± 10%	± 10%	± 10%	± 10%

Table 14. **The success criteria in Arnold.**

B.4. Detailed experimental results

As shown in the Table 10, 11 and 13, these six tables serve as a complement to the results in the experimental section of the article as an illustration of the validity of the model.

B.5. Success criteria

The success ranges of Arnold are shown in the table 14, and we verify the accuracy of the model by taking 1/5, 3/5, and 1 of the success ranges in the ablation experiment section.

B.6. Examples description

We'll use the example to illustrate the process of the model in inference. First, we assume that the training stage includes three base target states (0%, 50%, and 100%) and initializes three embeddings. When the training is completed, these three embeddings can represent the state features. In inference, for the new language "open drawer 60%", the LLMs analyze the instruction to get state value 60% and decompose it into 2 closest states (50% and 100%) with coefficients 0.8 and 0.2, then retrieve and combines learned embeddings to generate the 60% state embedding, achieving generalization to the novel state. In this way, the diffusion model can generate novel-state action.