

## 436 A Proof of Proposition 1

437 *Proof.* Consider the observation  $\mathbf{o}$  and the action  $\mathbf{a} = \pi(\mathbf{o})$ , let  $\mathbf{a}^k = \mathbf{a} + \varepsilon^k$ . By the definition of  
438 the noise prediction function  $\varepsilon$ , we have

$$\varepsilon^k = \varepsilon(\mathbf{o}, \pi(\mathbf{o}) + \varepsilon^k, k). \quad (2)$$

439 Applying  $g \in \text{SO}(2)$  to  $\mathbf{o}$  we have

$$\varepsilon^k = \varepsilon(g\mathbf{o}, \pi(g\mathbf{o}) + \varepsilon^k, k). \quad (3)$$

440 Since  $\pi$  is equivariant,

$$\varepsilon^k = \varepsilon(g\mathbf{o}, g\mathbf{a} + \varepsilon^k, k). \quad (4)$$

441 Since the noise prediction function predicts the noise as long as  $\varepsilon^k$  is the same on both sides of the  
442 equation, we can substitute  $\varepsilon^k$  with  $g\varepsilon^k$

$$g\varepsilon^k = \varepsilon(g\mathbf{o}, g\mathbf{a} + g\varepsilon^k, k). \quad (5)$$

443 By linearity,  $g\mathbf{a} + g\varepsilon^k = g(\mathbf{a} + \varepsilon^k) = g\mathbf{a}^k$  and thus

$$g\varepsilon^k = \varepsilon(g\mathbf{o}, g\mathbf{a}^k, k). \quad (6)$$

444 Replacing  $\varepsilon^k$  with  $\varepsilon(\mathbf{o}, \mathbf{a}^k, k)$  gives  $g\varepsilon(\mathbf{o}, \mathbf{a}^k, k) = \varepsilon(g\mathbf{o}, g\mathbf{a}^k, k)$  as desired.  $\square$

## 445 B Decomposing Group Representation in Relative Pose Control into 446 Irreducible Representations

447 In Section 4.2, we want to find a linear action  $\rho_{\mathbf{A}}$  that satisfies  $g\mathbf{a}_t = \rho_{\mathbf{A}}(g)\text{Vec}_r(\mathbf{A}_t) =$   
448  $\text{Vec}_r(T_g\mathbf{A}_tT_g^{-1})$ . solving for  $\rho_{\mathbf{A}} \in \mathbb{R}^{16 \times 16}$  we have the group action of  $\text{SO}(2)$  on  $\text{Vec}_r(\mathbf{A}_t)$   
449 as

$$\rho_{\mathbf{A}} = \begin{bmatrix} c^2 & -\frac{s_2}{2} & 0 & 0 & -\frac{s_2}{2} & s^2 & 0 & 0 \\ \frac{s_2}{2} & c^2 & 0 & 0 & -\frac{s_2}{2} & -\frac{s_2}{2} & 0 & 0 \\ 0 & 0 & c & 0 & 0 & 0 & -s & 0 \\ 0 & 0 & 0 & c & 0 & 0 & 0 & -s \\ \frac{s_2}{2} & -s^2 & 0 & 0 & c^2 & -\frac{s_2}{2} & 0 & 0 \\ s^2 & \frac{s_2}{2} & 0 & 0 & \frac{s_2}{2} & c^2 & 0 & 0 \\ 0 & 0 & s & 0 & 0 & 0 & c & 0 \\ 0 & 0 & 0 & s & 0 & 0 & 0 & c \end{bmatrix}, \quad (7)$$

$\begin{matrix} \rho_1(g) & & & \\ & I_2 & & \\ & & \rho_1(g) & \\ & & & I_2 \end{matrix}$

450 where  $c = \cos g$ ,  $s = \sin g$ ,  $c_2 = \cos 2g$ ,  $s_2 = \sin 2g$ . Define  $P$  as

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (8)$$

451 We then have

$$P\rho_{\mathbf{A}}P^{-1} = \begin{bmatrix} I_6 & & & & & \\ & \rho_1(g) & & & & \\ & & \rho_1(g) & & & \\ & & & \rho_1(g) & & \\ & & & & \rho_1(g) & \\ & & & & & \rho_2(g) \end{bmatrix}. \quad (9)$$

## 452 C Simplifying Group Action in Relative Pose Control

453 In Section 4.2, we want to find a linear action  $\rho_{\mathbf{A}}$  that satisfies

$$\rho_{\mathbf{A}}(g)\text{Vec}_r(\mathbf{A}_t) = \text{Vec}_r(T_g\mathbf{A}_tT_g^{-1}), \quad (10)$$

454 To simplify the problem, we decompose  $\mathbf{A}_t = \begin{bmatrix} \mathbf{R}_t & \mathbf{D}_t \\ 0 & 1 \end{bmatrix}$  where  $\mathbf{R}_t$  is the  $\text{SO}(3)$  rotation and  $\mathbf{D}_t =$   
 455  $[x, y, z]^T$  is the translation. Define  $R_g$  as the rotation matrix in  $T_g$ ,

$$R_g = \begin{bmatrix} \cos g & -\sin g & 0 \\ \sin g & \cos g & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \rho_1(g) & \\ & \rho_0(g) \end{bmatrix}. \quad (11)$$

456 Since the conjugate does not apply to translation, we can write  $g\mathbf{D}_t = R_g\mathbf{D}_t = [\rho_1(x, y), \rho_0(z)]^T$

457 For rotation, similar as before, we need to find the representation  $\rho_{\mathbf{R}}$  that satisfies

$$\rho_{\mathbf{R}}(g)\text{Vec}_r(\mathbf{R}_t) = \text{Vec}_r(R_g\mathbf{R}_tR_g^{-1}). \quad (12)$$

458 Solving for  $\rho_{\mathbf{R}}(g) \in \mathbb{R}^{9 \times 9}$  we have

$$\rho_{\mathbf{R}} = \begin{bmatrix} c^2 & -cs & 0 & -cs & s^2 & 0 & 0 & 0 & 0 \\ cs & c^2 & 0 & -s^2 & -cs & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & 0 & -s & 0 & 0 & 0 \\ cs & -s^2 & 0 & c^2 & -cs & 0 & 0 & 0 & 0 \\ s^2 & cs & 0 & cs & c^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & s & 0 & 0 & c & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c & -s & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (13)$$

459 where  $c = \cos g, s = \sin g$ . To decompose it into the irreducible representations of  $\text{SO}(2)$ , we  
 460 define

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (14)$$

461 Such that  $P\rho_{\mathbf{R}}P^{-1}$  is a block diagonal matrix consisting of irreducible representations

$$P\rho_{\mathbf{R}}P^{-1} = \begin{bmatrix} \rho_0(g) & & & & & \\ & \rho_0(g) & & & & \\ & & \rho_0(g) & & & \\ & & & \rho_1(g) & & \\ & & & & \rho_1(g) & \\ & & & & & \rho_2(g) \end{bmatrix}. \quad (15)$$

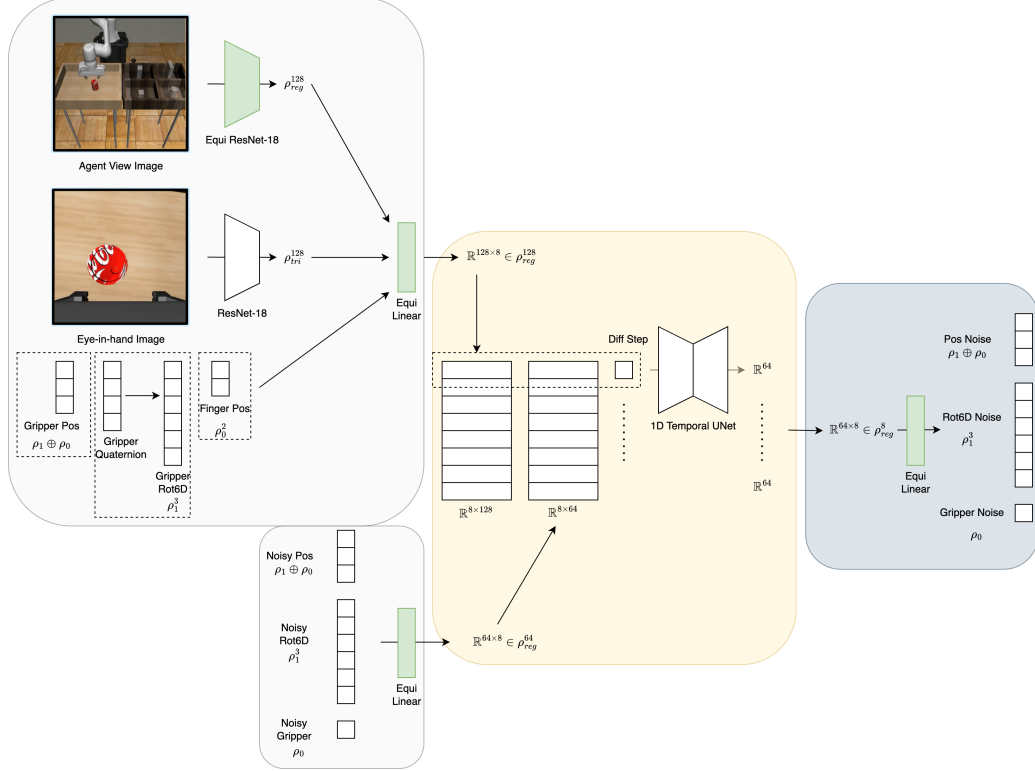


Figure 7: The detailed network architecture of our Equivariant Diffusion Policy in the simulation experiments.

We can then use  $\rho(g) = P\rho_{\mathbf{R}}P^{-1} = \rho_0^3(g) \oplus \rho_1^2(g) \oplus \rho_2(g) \in \mathbb{R}^{9 \times 9}$  as the group representation of the output of the equivariant network, then construct the  $3 \times 3$  rotation matrix  $\mathbf{R}_t$  using  $P$ . Specifically, let  $V \in \mathbb{R}^9$  be the output of the network associated with the representation  $\rho(g)$  (i.e.,  $g$  acts on  $V$  through  $\rho(g)V$ ). Define

$$\text{Vec}_r(\mathbf{R}_t) = P^{-1}V. \quad (16)$$

Applying  $\rho(g)$  on  $V$  will lead to

$$P^{-1}\rho(g)V \quad (17)$$

$$= P^{-1}P\rho_{\mathbf{R}}P^{-1}V \quad (18)$$

$$= \rho_{\mathbf{R}}\text{Vec}_r(\mathbf{R}), \quad (19)$$

which is the desired property in the equivariant network.

In the end, adding the group action for the translation ( $\rho_1 \oplus \rho_0$ ) and gripper open width ( $\rho_0$ ), we have  $\rho_a = \rho_0^5 \oplus \rho_1^3 \oplus \rho_2$ .

## D Network Architecture Detail

In the image version, we implement the equivariant observation encoder with an equivariant ResNet [51] for the agent view image, a standard ResNet [52] for the eye-in-hand image, and an equivariant MLP for the robot states. We implement the equivariant layers in the group  $C_8$ . Figure 7 shows the detailed network architecture of our Equivariant Diffusion Policy in the simulation experiments. The network is defined under group  $C_8$ . First, in the encoding phase, the agent view image is processed with an equivariant ResNet-18, whose output is a  $128 \times 8$ -dimensional regular representation vector of group. A non-equivariant ResNet-18 with a spatial maxpool at the end processes the eye-in-hand image and outputs a 128-dimensional representation vector that uses the trivial invariance representation. Those two vectors are concatenated with the gripper position (represented

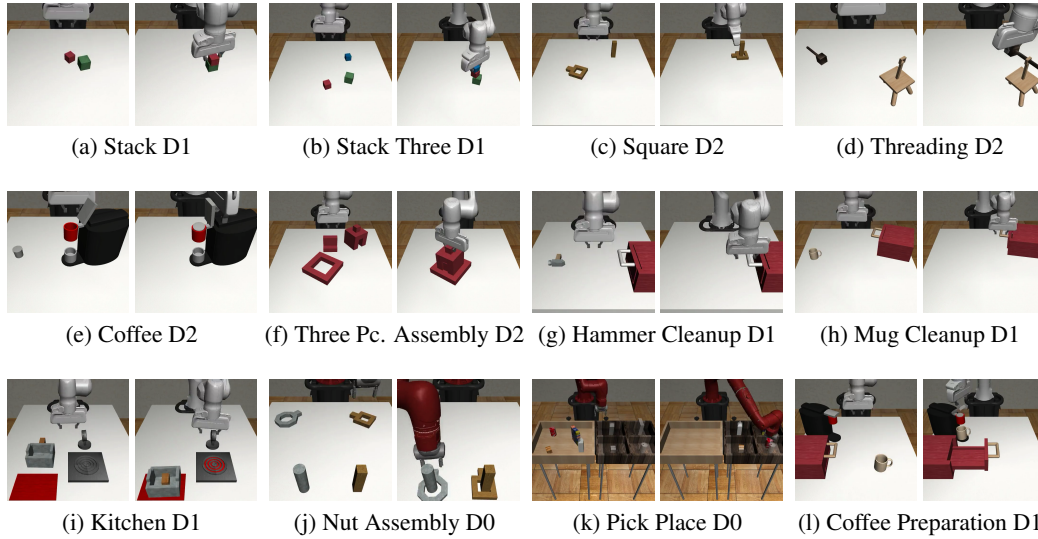


Figure 8: The experimental environments from MimicGen [11]. The left image in each sub figure shows an initial state of the environment; the right image shows the goal state.

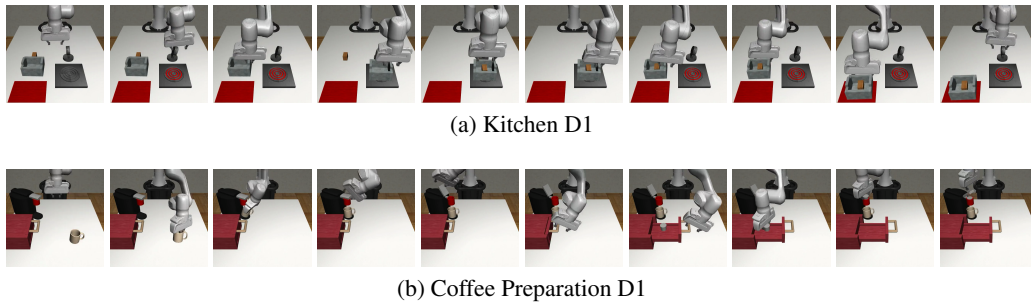


Figure 9: Illustration of an episode of Kitchen D1 task and Coffee Preparation D1 task.

480 using  $\rho_1 \oplus \rho_0$ ), gripper orientation (in the format of 6D rotation, represented using  $\rho_1^3$ ), and the  
481 gripper finger position (represented using  $\rho_0^2$ ). The concatenated mixed-representation vector is sent  
482 to an equivariant linear layer, whose output is a  $128 \times 8$ -dimensional regular representation obser-  
483 vation embedding. The noisy action is also encoded using an equivariant linear layer, whose output  
484 is a  $64 \times 8$ -dimensional regular representation action embedding. Second, in the denoising phase,  
485 we process each part of the observation embedding and the action embedding that corresponds to  
486 the same group element with a 1D Temporal UNet with hidden dimensions of [512, 1024, 2048] to  
487 get a 64-dimensional vector. Doing so for each pair, we will recover a  $64 \times 8$ -dimensional regular  
488 representation noise embedding. In the end, an equivariant linear layer will decode the noise.

489 In the voxel version, the agent view image is replaced with a voxel grid, and we replace the equiv-  
490 ariant ResNet with an 8-layer 3D equivariant convolutional encoder. The 1D Temporal UNet has a  
491 hidden dimensions of [256, 512, 1024]. The other part of the network stays the same. In the real-  
492 world, we remove the eye-in-hand image and only use the voxel grid as vision input (the gripper  
493 state vector stays the same).

## 494 E Simulation Environments

495 Figure 8 shows the initial and goal states of each tasks. Figure 9 shows an example trajectory for  
496 finishing the Kitchen and Coffee Preparation tasks. The RGB observation is an agent-view image  
497 and an eye-in-hand image with a size of  $3 \times 84 \times 84$ . The voxel grid observation has a size of



Task	Max Steps	Max Out of Plane Rot in Demo
Stack D1	400	11.2
Stack Three D1	400	13.2
Square D2	400	14.7
Threading D2	400	13.4
Coffee D2	400	14.1
Three Piece Assembly D2	500	16.2
Hammer Cleanup D1	500	16.4
Mug Cleanup D1	500	13.0
Kitchen D1	800	16.2
Nut Assembly D0	500	15.5
Pick Place D0	1000	18.0
Coffee Preparation D1	800	59.0

Table 4: The maximum number of time steps and the maximum out of plane rotation (in degrees) in the demo for each simulation environments. The maximum out of plane rotation in the demo is the maximum angular difference between the SO(3) rotation and the SO(2) rotation (i.e., only rotating around the  $z$  axis) over all demonstration steps, averaged over 1000 demonstration episodes.

Method	Obs	Obs Step	Action Pred. Step	Action Exec. Step
Equi. DiffPo (Vo)	Voxel Grid, Eye-In-Hand Image, Gripper State	1	16	8
Equi. DiffPo (Im)	Agent View Image, Eye-In-Hand Image, Gripper State	2	16	8
DiffPo-C	Agent View Image, Eye-In-Hand Image, Gripper State	2	16	8
DiffPo-T	Agent View Image, Eye-In-Hand Image, Gripper State	2	10	8
DP3	Point Cloud, Gripper State	2	16	8
ACT	Agent View Image, Eye-In-Hand Image, Gripper State	1	10	10
BC-RNN	Agent View Image, Eye-In-Hand Image, Gripper State	1	1	1

Table 5: The observation format, observation step, action prediction step, and action execution step for all methods. The gripper state is a vector including a 3 dimensional position vector, a rotation vector in the format of 6D rotation representation or 4D quaternion, and a 2 dimensional finger position.

498  $4 \times 64 \times 64 \times 64$  where the first channel is binary occupancy and the remaining three channels are  
499 RGB. The point cloud observation has a size of  $1024 \times 6$  (i.e, xyzrgb). The point cloud only contains  
500 points above the table, as suggested in [20]. All tasks have a full 6 DoF SE(3) action space. Table 4  
501 shows the maximum number of time steps (following [11]) and the maximum out of plane rotation  
502 in the demo, calculated by taking the maximum angular difference between the SO(3) rotation and  
503 the SO(2) rotation (i.e., only rotating around the  $z$  axis) across the entire demonstration episode.  
504 Results averaged for 1000 demonstrations.

## 505 F Training Detail

506 In the simulation experiments, we follow the hyper-parameters of the prior work [1] for the image  
507 version of our method, where the only change is that we increase the batch size to 128 for faster  
508 training. Specifically, the observation contains two steps of history observation, and the output of  
509 the denoising process is a sequence of 16 action steps. We use all 16 steps for training but only  
510 execute eight steps in evaluation. We train our models with the AdamW [53] optimizer (with a  
511 learning rate of  $10^{-4}$  and weight decay of  $10^{-6}$ ) and Exponential Moving Average (EMA). We use  
512 a cosine learning rate scheduler with 500 warm-up steps. We use DDPM [14] with 100 denoising  
513 steps for both training and evaluation. For each different number of demos (100, 200, 1000), we  
514 maintain roughly the same number of training steps by the training for  $50000/n$  epochs where  $n$  is  
515 the number of demos. Evaluations are conducted every  $1000/n$  epochs (50 evaluations in total). In  
516 the voxel version, we use only one step of history observation, and keep the other hyper-parameters  
517 the same.

518 The hyper-parameters for the diffusion policy and BC RNN baselines exactly follow [1]. We follow  
519 the original work [20] for the hyper-parameters of DP3, except that we use the same action sequence  
520 length (16 for training and 8 for evaluation) as [1] and our method. For the ACT baseline, we follow  
521 the hyper-parameters provided in the prior work [45], except that we use a chunk size of 10, KL

Ablation	Method	Ctrl	Obs	Stack D1			Stack Three D1			Square D2			Threading D2		
				100	200	1000	100	200	1000	100	200	1000	100	200	1000
-	Equi. DiffPo (Vo)	Abs	Voxel	99	100	100	75	91	91	39	48	63	39	53	55
No Voxel	Equi. DiffPo (Im)		RGB	93	100	100	55	77	96	25	41	60	22	40	59
No Equi.	DiffPo-C (Vo)		Voxel	87	99	100	33	79	94	10	24	60	19	43	54
No Voxel No Equi.	DiffPo-C [1]		RGB	76	97	100	38	72	94	8	19	46	17	35	59
Ablation	Method	Ctrl	Obs	Coffee D2			Three Pc. Asse. D2			Hammer Cleanup D1			Mug Cleanup D1		
				100	200	1000	100	200	1000	100	200	1000	100	200	1000
-	Equi. DiffPo (Vo)	Abs	Voxel	65	73	76	37	58	71	70	66	73	53	65	68
No Voxel	Equi. DiffPo (Im)		RGB	60	79	76	15	39	69	65	63	77	49	64	67
No Equi.	DiffPo-C (Vo)		Voxel	50	72	75	2	5	50	54	64	76	47	58	66
No Voxel No Equi.	DiffPo-C [1]		RGB	44	66	79	4	6	30	52	59	73	43	59	65
Ablation	Method	Ctrl	Obs	Kitchen D1			Nut Assembly D0			Pick Place D0			Coffee Prep. D1		
				100	200	1000	100	200	1000	100	200	1000	100	200	1000
-	Equi. DiffPo (Vo)	Abs	Voxel	85	89	88	67	77	83	58	69	82	80	83	85
No Voxel	Equi. DiffPo (Im)		RGB	67	77	81	74	85	94	42	74	92	77	83	85
No Equi.	DiffPo-C (Vo)		Voxel	82	87	87	66	77	84	41	67	84	65	75	77
No Voxel No Equi.	DiffPo-C [1]		RGB	67	85	87	55	68	83	35	65	83	65	62	58

Table 6: The ablation study that ablates the voxel input and the equivariant structure in our method. We experiment with 100, 200, and 1000 demos in each environment and report the maximum task success rate among 50 evaluations throughout training. Results averaged over three seeds.

Ablation	Method	Ctrl	Average over 12 Environments		
			100	200	1000
-	Equi. DiffPo (Vo)	Abs	63.9	72.6	77.9
No Voxel	Equi. DiffPo (Im)		53.7 (-10.3)	68.5 (-4.1)	79.7 (+1.8)
No Equi.	DiffPo-C (Vo)		46.3 (-17.6)	62.5 (-10.1)	75.6 (-2.3)
No Voxel No Equi.	DiffPo-C [1]		42.0 (-21.9)	57.8 (-14.8)	71.4 (-6.5)

Table 7: The average performance over 12 tasks of the ablation study. Number in parenthesis shows the performance difference after removing different components in our Equivariant Diffusion Policy with voxel input.

weight of 10, batch size of 64 with learning rate of  $5 \times 10^{-5}$ , and no temporal aggregation, following the tuning tips provided by the authors. See Table 5 for the observation format, observation step, action prediction step, and action execution step for all methods.

In the real-world experiments, we use a batch size of 64, one step of observation, and disable the EMA. We use DDIM [49] with 100 denoising steps for training and 16 denoising steps for evaluation. The other hyper-parameters stay the same as in simulation.

## G Ablation Study

We perform an ablation study regarding the equivariant structure and the voxel input in our method. We consider the following four candidates: 1) Ours: our Equivariant Diffusion Policy with voxel input; 2) Ours no Voxel: our Equivariant Diffusion Policy with RGB input; 3) Ours no Equi.: the baseline Diffusion Policy with voxel input; 4) Ours no Voxel no Equi.: the baseline Diffusion Policy with RGB input, same as [1]. Table 6 shows the result and Table 7 shows the average over all 12 environments. Though both the equivariant structure and the voxel input contribute to the performance improvement of our method, the equivariant structure plays a more important rule, as removing it (No Equi.) lead to a more significant performance drop compared with removing the voxel input (No Voxel). Note that by using the voxel input, Diffpo-C (Vo) is marginally better than the original Diffusion Policy (DiffPo-C), thus we use Diffpo-C (Vo) as the baseline in our robot experiment in Section 5.3.

	Stack Three D1	Threading D2	Coffee Preparation D1
Equi. DiffPo (Im), SE(3) Action	77.3	40.0	85.3
Equi. DiffPo (Im), SE(2) Action	75.3	12.7	0.0

Table 8: Performance of Equivariant Diffusion Policy in SE(2) action space compared with SE(3) action space. 200 demos are used in this experiment.

## H SE(2) Action Space Variation

In this section, we evaluate a variation of our Equivariant Diffusion Policy in an SE(2) (with  $z$  translation) action space to demonstrate the necessity of leveraging an SE(3) action space. Specifically, the SE(2) agent only learns the top-down rotation and the out-of-plane rotations will be constantly set to 0. As is shown in Table 8, the SE(2) variation achieves a similar performance as the SE(3) version in Stack Three, as the demonstration data in this task has the least amount of out-of-plane rotation (as shown in Table 4). On the other hand, the SE(2) variation significantly underperforms in Threading, since the ability of wiggling the out-of-plane rotation helps the agent to precisely insert the tool. In the end, the SE(2) agent cannot solve Coffee Preparation at all, because the task requires a significant amount of out-of-plane rotation (as shown in Figure 9b).

## I Real-Robot Environment Details

Figure 6 shows the five tasks in this experiment. In Oven Opening, the oven is randomly initialized at one of the four borders of the workspace. In Banana in Bowl, the initial poses of the banana and the bowl are both randomly sampled. In Trash Sweeping, the robot needs to use a tool brush to sweep two pieces of crumpled paper out of its workspace. The initial poses of the objects are randomly sampled. In Letter Alignment, the robot needs to align the letters to form “AI”. The letter A is randomly initialized at one of the four corners of the workspace, and the pose of the I is randomly sampled. In Hammer to Drawer, the robot needs to open a drawer, pick up a hammer, place it inside the drawer, and close the drawer. The drawer is initialized at one of the four borders of the workspace, and the hammer is randomly initialized at the opposite side of the drawer. Lastly, we also evaluate a Bagel Baking task with an extremely long time horizon, where the robot needs to open the oven, pull out the tray inside the oven, pick up the bagel, place it inside the tray, close the tray, and close the oven. In this task, the oven is randomly initialized at one of the three borders of the workspace (where we eliminate the side that is furthest from the robot to avoid joint limits of the robot), and the bagel is randomly initialized along the opposite side of the oven. The observation is a voxel grid with a resolution of  $64 \times 64 \times 64$  and the gripper pose and open width. The voxel grid covers the  $(0.4m)^3$  workspace. During training, we apply a random crop augmentation to crop the voxel grid to  $58 \times 58 \times 58$ . In Banana in Bowl and Trash Sweeping, we train the model with an additional random rotation augmentation. The baseline is trained with the same data augmentation as our method.