

Appendix for “Equivariant Motion Manifold Primitives”

Anonymous Author(s)

Affiliation

Address

email

1 A Proofs of Propositions

2 *Proof of Group-Equivariance of (3.1).* The equivariance of the ground truth EMMP and the invari-
3 ance of g proves the proposition as follows:

$$\mathcal{Z}_{h \cdot \tau} = g(\mathcal{M}_{h \cdot \tau}, h \cdot \tau) = g\left(\bigcup_{x \in \mathcal{M}_\tau} h \cdot (x, \tau)\right) = g\left(\bigcup_{x \in \mathcal{M}_\tau} (x, \tau)\right) = g(\mathcal{M}_\tau, \tau) = \mathcal{Z}_\tau. \quad (1)$$

4 □

5 *Proof of Group-Equivariance of (3.2).* The equivariance of f and invariance of g proves the propo-
6 sition as follows:

$$\widehat{\mathcal{M}}_{h \cdot \tau} = f(\mathcal{Z}_{h \cdot \tau}, h \cdot \tau) = \bigcup_{z \in \mathcal{Z}_\tau} [h \cdot (f(z, \tau), \tau)]_x = \bigcup_{x \in \widehat{\mathcal{M}}_\tau} [h \cdot (x, \tau)]_x = \{[h \cdot (x, \tau)]_x \mid x \in \widehat{\mathcal{M}}_\tau\}. \quad (2)$$

7 □

8 *Proof of Group-Equivariance of (3.3).* Invariance of g_ϕ can be seen by the equivariance of \bar{h} , as
9 follows:

$$g_\phi(h \cdot (x, \tau)) = G_\phi(\bar{h}(h \cdot \tau)^{-1} \cdot (h \cdot (x, \tau))) = G_\phi((h\bar{h}(\tau))^{-1}h) \cdot (x, \tau) = G_\phi(\bar{h}(\tau))^{-1} \cdot (x, \tau) = g_\phi(x, \tau) \quad (3)$$

10 Equivariance of f_θ can be seen as follows:

$$\begin{aligned} f_\theta(z, h \cdot \tau) &= [\bar{h}(h \cdot \tau) \cdot (F_\theta(z, \bar{h}(h \cdot \tau)^{-1} \cdot (h \cdot \tau))), \bar{h}(h \cdot \tau)^{-1} \cdot (h \cdot \tau)]_x \\ &= [h \cdot \bar{h}(\tau) \cdot (F_\theta(z, \bar{h}(\tau)^{-1}(\tau)), \bar{h}(\tau)^{-1} \cdot \tau)]_x \\ &= [h \cdot (f_\theta(z, \tau), \tau)]_x \end{aligned} \quad (4)$$

11 □

12 B Related Works

13 In this section, we provide an overview of areas related to our work.

14 B.1 Movement Primitives

15 In this section, we consider any form of mathematical representation that describes motions (e.g.,
16 trajectories) as movement primitives.

17 Motion primitives as generative models that output motion trajectories in any form, given a param-
18 eter that specifies the task, which we refer to as a task parameter. The goal of motion primitives
19 is to generate motion trajectories similar to demonstration trajectories given by humans. One stan-
20 dard way @@ The generated trajectories of movement primitives can be parameterized and repre-
21 sented in various forms. One standard way is to represent the trajectories using a stable dynamical
22 system. A dynamical system can be seen as a movement primitive since it can output a solution
23 trajectory given an initial state. Dynamic movement primitives represent trajectories in the form
24 of time-dependent nonlinear dynamical systems which consist of mass-spring-damper systems with

25 additional force term [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Stable dynamical system-based approaches encode
 26 demonstration trajectories into state-dependent dynamical systems that are globally asymptotically
 27 stable [11, 12, 13, 14, 15, 16, 17, 18, 19].

28 However, the task parameters of most movement primitives are strictly restricted to the initial and
 29 final configurations. This limits the range of tasks that can be parameterized. In the case of water
 30 pouring, the cup’s position and the amount of water can not be represented by the initial and final
 31 configurations. By adopting conditional variational autoencoder’s structure, MMPs and EMMPs
 32 provide freedom of defining task parameters.

33 **B.2 Autoencoder-Based Manifold Learning**

34 Autoencoders have gained prominence in recent years for identifying and generating samples from
 35 a given data distribution’s underlying low-dimensional manifold structure. The main reason autoen-
 36 coders are frequently adopted for manifold learning is that they learn the latent space coordinates
 37 along with the manifolds. To learn more accurate manifolds, researchers have introduced addi-
 38 tional regularization terms [20, 21, 22, 23, 24]. For a specific structure of conditional variational
 39 autoencoder, where the decoder gets an additional conditional parameter, the need to disentangle the
 40 conditional inputs and latent values has risen. [25] introduced a regularization term to disentangle
 41 input spaces of its decoder, by solving adding an auxiliary neural network to estimate conditional
 42 inputs from latent values, and regularizing the autoencoder by making it harder for the auxiliary
 43 network to estimate. However, unlike the independence regularization term that we introduced, the
 44 regularization term does not necessarily guarantee independence between the two spaces.

45 **B.2.1 Autoencoder-Based Motion Manifold primitive**

46 In this section, we introduce an existing motion manifold primitive framework called TC-VAE [26].
 47 TC-VAE aims to parameterize the motion manifold given a task parameter based on autoencoder
 48 frameworks. As TC-VAE adopts the structure of [25], the decoder of it takes additional task param-
 49 eter inputs other than the latent value inputs. TC-VAE also adopts the regularization term of [25]
 50 for disentangling the task parameters and the latent values, which still shares the shortcoming of not
 51 guaranteeing independence between latent space and the conditional input space.

52 **B.3 Equivariant Models in Robotics**

53 Invariance and equivariance properties have played a role in deep learning models as an inductive
 54 bias to generalize well and be trained data efficiently [27]. Translation equivariance in convolu-
 55 tional neural networks (CNNs) has been effective for image recognition tasks [28]. Group equiv-
 56 ariant CNNs have expanded the equivariance in CNNs to more complex equivariance, e.g. SO(3)-
 57 equivariance achieved by spherical CNNs [29, 30]. In robot manipulation tasks, [31] proposed an
 58 SE(3)-equivariant object representation, and [32] introduced a SE(2)-equivariant dynamics model
 59 learning for pushing manipulation. Most of the existing equivariant models in robotics are restricted
 60 to certain types of groups, whereas our work can be applied to tasks with arbitrary group symmetries.

61 **C Experimental and Implementation Details**

62 Throughout the experiments, we have used RTX 2080 Ti, RTX 3080 Ti, RTX 3090 for training the
 63 models, and each experiment takes a few hours to 10 hours depending on the model.

64 **C.1 Evaluation Metrics**

65 **Reconstruction Error:** We measure reconstruction error in the test dataset using following equation:

$$\text{Reconstruction loss} = \sqrt{\frac{1}{N} \sum_{i=1}^N \frac{1}{M_i} \sum_{j=1}^{M_i} \frac{1}{T} d_{\mathcal{X}}^2(f_{\theta}(g_{\phi}(x_{ij}, \tau_i), \tau_i), x_{ij})}. \quad (5)$$

66 **Latent-Task Dataset:** To calculate mutual information and negative log-likelihood, we define a
 67 dataset of (z, τ) paired dataset. To build the dataset large enough, we first randomly augment (x, τ)
 68 pairs in the training dataset 100 times. Then, the every z is the encoded values from (x, τ) ; $z =$
 69 $g_\phi(x, \tau)$.

70 **Mutual Information:** Mutual information between z and τ is measured using Mutual Informa-
 71 tion Neural Estimator (MINE) [33]. MINE estimates by, which estimates mutual information by
 72 maximizing its lower bound, using the Donsker-Varadhan representation:

$$D_{KL}(\mathcal{Z}||\mathcal{T}) \geq \sup_{F \in \mathcal{F}} \mathbb{E}_{\mathcal{Z}}[F] - \log(\mathbb{E}_{\mathcal{T}}e^F), \quad (6)$$

73 where \mathcal{F} is any class of functions $F : \Omega \rightarrow \mathbb{R}$. In our case, $\Omega = \mathcal{Z} \times \mathcal{T}$. By replacing \mathcal{F} by
 74 parametric family \mathcal{F}_Θ , the mutual information is estimated as follows:

$$I_\Theta(\mathcal{Z}, \mathcal{T}) = \sup_{\theta \in \Theta} \mathbb{E}_{p(z, \tau)} F_\theta - \log(\mathbb{E}_{p(z) p(\tau)} e^{F_\theta}). \quad (7)$$

75 We train MINE using the latent-task dataset for 1,500 iterations with a batch size of 5,000 equally
 76 for all models.

77 **Negative Log-Likelihood:** Given (z, τ) from the latent-task dataset, we calculate negative log-
 78 likelihood $-\log(p_{\mathcal{M}_\tau}(g_\phi(z)))$ in trajectory space \mathcal{X} , using the following equation:

$$p_{\mathcal{M}_\tau}(g_\phi(z, \tau)) = p_{\mathcal{Z}_\tau}(z) |\det [J_{g_\phi}^T J_{g_\phi}]|^{-\frac{1}{2}}, \quad (8)$$

79 where J_{g_ϕ} denotes $\frac{\partial g_\phi}{\partial z}(z, \tau)$.

80 C.2 Planar Mobile Robot Experiment

81 C.2.1 Formulas and Proofs

82 **Task Parameter Space:** Recall that the configuration space is $\mathcal{Q} = \mathbb{R}^2$, the trajectory space is
 83 $\mathcal{X} = \mathbb{R}^{2T}$. The task parameter τ consists of the initial position of the mobile robot, denoted by
 84 (q_r^1, q_r^2) , and the rotated angle of the wall axis \hat{x}_w with respect to \hat{x}_s , denoted by ω_w .

85 To uniformly sample from \mathcal{T} we make \mathcal{T} compact by restricting the initial position of the motile
 86 robot to be on a disk whose center is the origin, the inner radius is 5, and the outer radius is 10.
 87 Also, we have limited the wall rotation axis to $-\frac{\pi}{4} \leq \omega_w < \frac{\pi}{4}$ to make the problem easier for
 88 non-equivariant methods e.g. TC-VAE. since the wall's geometries are identical every 90 degrees,
 89 the wall still can span all possible geometrical configurations.

90 **Trajectory Space and Distance Measure:** We set the length of trajectories $T = 201$, which makes
 91 the trajectory space $\mathcal{X} = \mathbb{R}^{402}$. The distance measure is defined as: $d_{\mathcal{X}}(x_1, x_2) := \|x_1, x_2\|_2$.

92 **Group Operations:** Recall that the symmetry group H is $H := p4m \times \text{SO}(2)$, where $p4m$ is
 93 a specific type of wallpaper group and $\text{SO}(2)$ is the group of 2×2 rotation matrices. We de-
 94 note $p4m := \{(i, j) | i \in \{0, 1\}, j \in \{0, 1, 2, 3\}\}$, where $i \in \{0, 1\}$ represents flipping and
 95 $j \in \{0, 1, 2, 3\}$ represents $n\pi/2$ rotation. Throughout this section, we represent an $\text{SO}(2)$ element

$$96 R = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \in \text{SO}(2) \text{ simply as } \alpha.$$

97 Given two group elements $((a, b), \alpha), ((c, d), \beta) \in p4m \times \text{SO}(2)$, the group operation is defined as:

$$((a, b), \alpha)((c, d), \beta) = ((\text{mod}(a + c, 2), \text{mod}(b + (-1)^a d, 4)), \alpha + \beta), \quad (9)$$

98 where $\text{mod}(x, y)$ denotes the remainder of $\frac{x}{y}$.

99 **Group Actions:** The procedure of the group actions of $H = p4m \times \text{SO}(2)$ can be explained as
 100 follows: (i) flip the robot over the wall axis (ii) rotate the robot around the origin $\frac{n\pi}{2}$, and finally,
 101 (iii) rotated the robot and the wall. Given a task parameter $\tau = (q_r^1, q_r^2, \omega_w)$ and a group element
 102 $h = ((a, b), \alpha)$, the group action $h \cdot \tau$ is then defined as:

$$h \cdot \tau = (\text{Rot}(\alpha + \omega_w + \frac{b\pi}{2}) * \text{flip}(a) * \text{Rot}(-\omega_w) * (q_r^1, q_r^2), \alpha + \omega_w) \quad (10)$$

103 where $*$ denotes matrix multiplication, $\text{flip}(a) := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}^a$, and $\text{Rot}(\alpha) := \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}$.

104 Given a trajectory $x = \{(q_i^1, q_i^2)\}_{i=1}^T$, a task parameter $\tau = (q_r^1, q_r^2, \omega_w)$ and a group element
 105 $h = ((a, b), \alpha)$, the group action $h \cdot (x, \tau)$ is defined as :

$$h \cdot (x, \tau) = (\{\text{Rot}(\alpha + \omega_w + \frac{b\pi}{2}) * (q_i^1, q_i^2)\}_{i=1}^T, h \cdot \tau). \quad (11)$$

106 **Group-Equivariant Module \bar{h} :** Given a task parameter τ , $\bar{h}(\tau)$ can be divided into two elements,
 107 $\bar{h}(\tau) = (\bar{h}_1(\tau), \bar{h}_2(\tau))$, where $\bar{h}_1(\tau) = (\bar{h}_1^1(\tau), \bar{h}_1^2(\tau)) \in p4m$ and $\bar{h}_2(\tau) \in \text{SO}(2)$. Figure 1
 108 illustrates $\bar{h}(\tau)$, and its equivariance for \bar{h}_1 using two group actions $h_{\text{flip}} = (1, 0, 0)$ and $h_{\text{rot90}} =$
 109 $(0, 1, 0)$. It can be seen by the commutative diagram that $\bar{h}(h_{\text{flip}} \cdot \tau) = h_{\text{flip}} \bar{h}(\tau)$ and $\bar{h}(h_{\text{rot90}} \cdot$
 110 $\tau) = h_{\text{rot90}} \bar{h}(\tau)$. The rest cases of flipping and rotating 90, 180, 270 degrees can be shown in the
 111 same way. As shown, $\bar{h}_2(\tau)$ is defined as $\bar{h}_2(\tau) = \omega_w$. The equivariance of \bar{h}_2 can be shown by
 112 $h_{\text{rot}} = (0, 0, \alpha)$ and $\tau = (q_r^1, q_r^2, \omega_w)$:

$$\bar{h}(h_{\text{rot}} \cdot \tau) = (q_r^1, q_r^2, \omega_w + \alpha) = (0, 0, \alpha)(q_r^1, q_r^2, \omega_w) = h_c \bar{h}(\tau). \quad (12)$$

113 Equivariance for the case where $h = (a, b, \alpha)$ is then simply shown by dividing it into $h =$
 114 $(0, 0, \alpha)(a, b, 0)$:

$$\begin{aligned} \bar{h}((a, b, \alpha) \cdot \tau) &= \bar{h}(((0, 0, \alpha)(a, b, 0)) \cdot \tau) \\ &= \bar{h}((0, 0, \alpha) \cdot (a, b, 0) \cdot \tau) \\ &= (0, 0, \alpha) \bar{h}((a, b, 0) \cdot \tau) \\ &= (a, b, 0)(0, 0, \alpha) \bar{h}(\tau) \\ &= (a, b, \alpha) \bar{h}(\tau). \end{aligned} \quad (13)$$

115 Below equation is the formal definition of \bar{h} given a $\tau = (q_r^1, q_r^2, \omega_w)$:

$$\begin{aligned} \bar{h}_1^1(\tau) &= \begin{cases} 0, & \text{if } \frac{k\pi}{2} \leq \text{atan2}(q_r^2, q_r^1) - \omega_w < \frac{k\pi}{2} + \frac{\pi}{4}, \\ 1, & \text{otherwise} \end{cases}, \\ \bar{h}_1^2(\tau) &= \begin{cases} 0, & \text{if } -\frac{\pi}{4} \leq \text{atan2}(q_r^2, q_r^1) - \omega_w < \frac{\pi}{4} \\ 1, & \text{if } \frac{\pi}{4} \leq \text{atan2}(q_r^2, q_r^1) - \omega_w < \frac{3\pi}{4} \\ 2, & \text{if } \frac{3\pi}{4} \leq \text{atan2}(q_r^2, q_r^1) - \omega_w < \pi \\ & \text{or } -\pi \leq \text{atan2}(q_r^2, q_r^1) - \omega_w < -\frac{3\pi}{4} \\ 3, & \text{if } -\frac{3\pi}{4} \leq \text{atan2}(q_r^2, q_r^1) - \omega_w < -\frac{\pi}{4} \end{cases}, \\ \bar{h}_2(\tau) &= \omega_w, \end{aligned} \quad (14)$$

116 where $k \in \{-2, -1, 0, 1\}$ and $(\text{atan2}(q_r^2, q_r^1) - \omega_w)$ is assumed to be satisfying $-\pi \leq$
 117 $\text{atan2}(q_r^2, q_r^1) - \omega_w < \pi$.

118 C.2.2 Experimental Details

119 **Datasets:** For dataset generation, we first uniformly sample from the smallest space that can span
 120 \mathcal{T} by symmetry transformations, in which the robot's initial position q_r satisfies $5 \leq \|q_r\| < 10$
 121 and $0 \leq \text{atan2}(q_r^2, q_r^1) < \pi/4$, and the wall axis angle is 0. We collect trajectory data by generating
 122 B-splines given via points labeled by humans. The B-splines are then reparameterized so that the
 123 time length of the splines becomes 5 seconds, where the splines accelerate for the first second and
 124 decelerate for the last second. We finally sample 201 points from the splines.

125 For the training dataset, we have gathered 6 trajectories for 75 randomly given task parameters, a
 126 total of 300 trajectories for training. For the validation dataset, we have gathered a trajectory for 80
 127 randomly given task parameters and randomly augmented them using symmetry transformations 100
 128 times. For the test dataset, we have gathered a trajectory for 40 randomly given task parameters and
 129 randomly augmented them using symmetry transformations 1000 times. The number of validation
 130 and test datasets are then 8,000 and 40,000 respectively.

131 **Network Architectures and Training Details:** A task parameter τ is represented as (q_r^1, q_r^2, ω_w) ,
 132 where (q_r^1, q_r^2) is the mobile robot's initial position and ω_w is the wall's axis angle. In practical

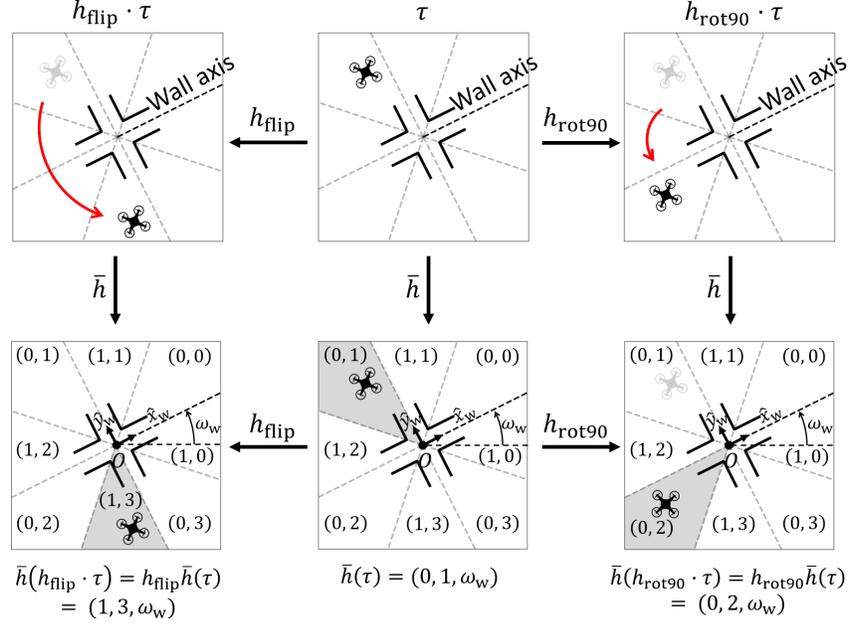


Figure 1: Illustration of \bar{h} and its equivariance. $h_{\text{flip}} := (1, 0, 0)$ is the flipping motion of the mobile robot over the wall axis, and $h_{\text{rot90}} := (0, 1, 0)$ is the rotation of the mobile robot 90 degrees around the origin. It can be seen that $h_{\text{flip}} \bar{h}(\tau) = (1, 0, 0)(0, 1, \omega_w) = (1, \text{mod}(0 - 1, 4), \omega_w) = (1, 3, \omega_w) = \bar{h}(h_{\text{flip}} \cdot \tau)$ and $h_{\text{rot90}} \bar{h}(\tau) = (0, 1, 0)(0, 1, \omega_w) = (0, \text{mod}(1 + 1, 4), \omega_w) = (0, 2, \omega_w) = \bar{h}(h_{\text{rot90}} \cdot \tau)$.

133 implementation, we use $(q_r^1, q_r^2, \cos \omega_w, \sin \omega_w) \in \mathbb{R}^4$ as an input parameter vector. Since $T = 201$,
 134 the output space is \mathbb{R}^{402} .

135 We use two-layer fully connected neural networks of 512 nodes for MMPs and EMMPs with elu
 136 as their activation function. TC-VAE’s encoder includes a fully connected network and a temporal
 137 convolutional network, and the decoder includes two fully connected networks for z and τ , a tem-
 138 poral convolutional network, and a fully connected network. All four fully connected networks used
 139 in TC-VAE are of two layers with size 434. The output sizes of fully connected networks for z and
 140 τ in the decoder are 36 and 72 respectively. The two temporal convolutional layers in TC-VAE are
 141 both with channel sizes (18, 36, 72) and kernel size 3. More details on the structure of TC-VAE are
 142 in [26]. All models in the experiments have similar number of parameters ($\approx 9.4 \times 10^5$).

143 **Success Criterion:** We consider a trajectory successful if it is consistent with the task parameter and
 144 reaches the goal without colliding with the wall. More specifically, we check (i) collision avoidance,
 145 (ii) the robot’s initial position, and the robot’s final position. We consider the trajectory satisfies (ii)
 146 and (iii) if the initial configuration and the final configuration are within a radius of 0.3 at the initial
 147 position specified in the task parameter and origin, respectively. The number of sample (z, τ) we
 148 use for success rate calculation is 50,000.

149 C.2.3 Additional results

150 **Architecture Comparison:** We compare MMPs and EMMPs of fully connected autoencoders (de-
 151 noted as AE), fully connected variational autoencoders (denoted as VAE), and variational autoen-
 152 coders of the same structure with TC-VAE (denoted as TC-VAE). Table 1 shows the four evaluation
 153 metrics. Overall, as shown in the success rate scores, regardless of network architecture and autoen-
 154 coder method, EMMPs without regularization perform the best, and MMPs without regularization
 155 perform the worst. Although EMMP (TC-VAE) excels in most measures (MI and NLL) its success
 156 rate (91.2%) is still lower than EMMP (AE)’s (92.40%) and EMMP (VAE)’s (95.72%), which is
 157 caused by the tendency of (TC-VAE) that it violates the initial and final condition in about 6% of
 158 trials, whereas EMMP (AE) only violates them and EMMP (VAE) almost never violate them (0%
 159 $\sim 0.01\%$).

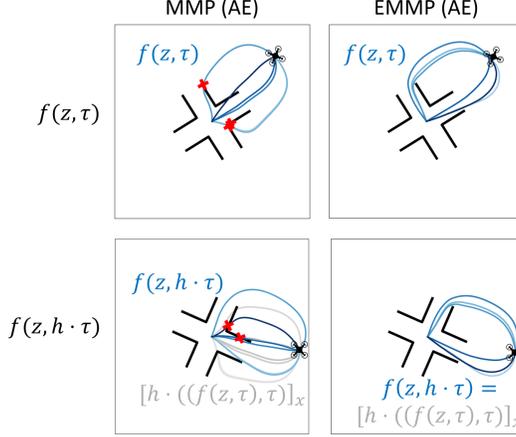


Figure 2: Equivariance comparison between MMP (AE) and EMMP (AE). If the decoder f is equivariant, $f(z, h \cdot \tau)$ (blue lines in the figure) and $[h \cdot (f(z, \tau), \tau)]_x$ (grey lines in the figure) must overlap. It can be seen that the decoder of MMP is not equivariant, whereas the EMMP’ decoder is equivariant.

160 **Equivariance Comparison:** Here, we qualitatively compare the equivariance performance of ran-
 161 dom data augmentation and equivariant learning method by comparing MMP (AE) and EMMP
 162 (AE). Figure 5 shows trajectories generated by τ and $h \cdot \tau$, with same z . If the decoder f is equivari-
 163 ant, $f(z, h \cdot \tau)$ (blue lines in the figure) and $[h \cdot (f(z, \tau), \tau)]_x$ (grey lines in the figure) must overlap.
 164 However, as shown in the Figure 5 *Left*, trajectories of the MMP do not , whereas trajectories of the
 165 EMMP perfectly overlap as shwon in Figure 5 *Right*.

166 C.3 Water-Pouring Experiment

167 C.3.1 Formulas and Proofs

168 **Task Parameter Space:** The input space $Q = \mathbb{R}^2 \times \text{SE}(3) \times [0.2, 0.41]$. A task parameter τ can be
 169 represented as $((q_c^1, q_c^2), (q_b^1, q_b^2, q_b^3, R_b), m_w)$, where (q_c^1, q_c^2) is the cup’s position, $(q_b^1, q_b^2, q_b^3, R_b)$
 170 is the bottle’s initial position and orientation, and m_w is the weight of water in the bottle. To
 171 construct compact \mathcal{T} , we limit (q_c^1, q_c^2) to be inside a square at the origin with edge length of 0.5,
 172 i.e., $-0.25 \leq q_c^1, q_c^2 \leq 0.25$, and limit the distance between (q_c^1, q_c^2) and (q_b^1, q_b^2) to satisfy $0.3 \leq$
 173 $\|q_b^1 - q_c^1, q_b^2 - q_c^2\|_2 \leq 0.78$. Since the bottle is on the table upright, q_b^3 is a constant.

174 **Trajectory Space and Distance Measure:** We set the length of trajectories T to be 480, which
 175 makes trajectory space $\mathcal{X} = \text{SE}(3)^{480}$. Given $x_1 = \{x_1^i\}_{i=1}^{480}$ and $x_2 = \{x_2^i\}_{i=1}^{480}$, where each x_j^i can
 176 be represented by $(R_{ij} \in \text{SO}(3), p_{ij} \in \mathbb{R}^3)$, the distance measure on \mathcal{X} is defined as:

$$d_{\mathcal{X}}(x_1, x_2) := \sqrt{\sum_i (\|R_{i1}^{-1} * R_{i2} - I\|_F^2 + \gamma \|p_{i1} - p_{i2}\|_2^2)}, \quad (15)$$

177 where $\gamma = 5$ is a constant.

Table 1: Reconstruction Error (RE), Mutual Information (MI), and Negative Log-Likelihood (NLL); the lower, the better. Success Rate (SR); the higher, the better.

Method	RE (↓)	MI (↓)	NLL (↓)	SR (↑)
MMP (AE)	0.223	0.487	1.49×10^4	50.08%
MMP (VAE)	0.233	0.687	1.57×10^4	42.98%
MMP (AE) + indep	0.225	0.329	1.48×10^4	52.39%
MMP (VAE) + indep	0.229	0.652	1.56×10^4	44.28%
EMMP (AE)	0.223	0.082	1.25×10^4	92.40%
EMMP (AE) + indep	0.229	0.077	1.24×10^4	86.66%
EMMP (VAE)	0.231	0.066	1.23×10^4	95.72%
EMMP (VAE) + indep	0.225	0.167	1.28×10^4	82.02%
EMMP (TC-VAE)	0.227	0.065	1.00×10^4	91.20%
EMMP (TC-VAE) + indep	0.247	0.071	1.15×10^4	88.22%

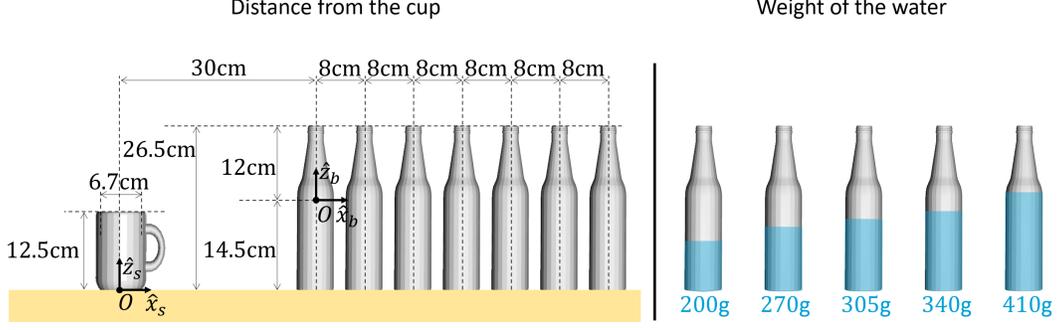


Figure 3: Task parameters for demonstration. The cup is always at the origin, and the bottle is at $(0, r \in \{0.3, 0.38, 0.54, 0.62, 0.78, 0.145\})$. The mass of water in the bottle is $m_w \in \{0.2, 0.27, 0.305, 0.34, 0.41\}$.

178 **Group Operations:** The symmetry group $H = \mathbb{R}^2 \times \text{SO}(2) \times \text{SO}(2)$, each is for translation of the
 179 cup and the bottle, rotation of the bottle around the cup, and rotation of the bottle around itself. Given
 180 two group elements $(a, b, R_{c1}, R_{b1}), (c, d, R_{c2}, R_{b2}) \in \mathbb{R}^2 \times \text{SO}(2) \times \text{SO}(2)$, the group operation
 181 is defined as follows:

$$(a, b, R_\alpha, R_\beta)(c, d, R_\gamma, R_\delta) = (a + c, b + d, R_\alpha * R_\gamma, R_\beta * R_\delta). \quad (16)$$

182 **Group Actions:** The procedure of group actions of $h \in H$ can be explained as follows: (i)
 183 translate the cup and the bottle, (ii) rotate the bottle around the cup, and (iii) rotate the bottle
 184 around itself. Given a task parameter $\tau = ((q_c^1, q_c^2), (q_b^1, q_b^2, q_b^3, R_b)), m_w$, and a group element
 185 $h = (a, b, R_\alpha, R_\beta)$, the group action $h \cdot \tau$ is defined as:

$$h \cdot \tau = ((q_c^1 + a, q_c^2 + b), ((q_c^1 + a, q_c^2 + b, 0) + (R_\alpha * (q_b^1 - q_c^1, q_b^2 - q_c^2, q_b^3)), R_\alpha * R_b * R_\beta), m_w). \quad (17)$$

186 **Group-Equivariant Module \bar{h} :** Given a task parameter $\tau = ((q_c^1, q_c^2), (q_b^1, q_b^2, q_b^3, R_b)), m_w$,
 187 $\bar{h}(\tau) = (\bar{h}_1(\tau), \bar{h}_2(\tau), \bar{h}_3(\tau))$ is defined as follows:

$$\bar{h}_1(\tau) = (q_c^1, q_c^2) \in \mathbb{R}^2, \quad (18)$$

$$\bar{h}_2(\tau) = \text{Rot}(\hat{z}, \theta_1), \quad (19)$$

$$\bar{h}_3(\tau) = \text{Rot}(\hat{z}, (\theta_2 - \theta_1)), \quad (20)$$

188 where $\theta_1 := \text{atan2}(q_b^2 - q_c^2, q_b^1 - q_c^1)$, $\theta_2 := \text{atan2}(\hat{x}_b^2, \hat{x}_b^1)$, and \hat{x}_b denotes the first column of R_b .

189 Given an arbitrary $h = (a, b, R_\alpha, R_\beta)$, where $R_\alpha = \text{Rot}(\hat{z}, \alpha)$ and $R_\beta = \text{Rot}(\hat{z}, \beta)$, the equivariance
 190 of \bar{h} is shown by the following equation:

$$\begin{aligned} \bar{h}(h \cdot \tau) &= \bar{h}((q_c^1 + a, q_c^2 + b), ((q_c^1 + a, q_c^2 + b, 0) + (R_\alpha * (q_b^1 - q_c^1, q_b^2 - q_c^2, q_b^3)), R_\alpha * R_b * R_\beta), m_w) \\ &= ((q_c^1 + a, q_c^2 + b), \text{Rot}(\hat{z}, \alpha + \theta_1), \text{Rot}(\theta_2 - \theta_1 + \beta)) \\ &= ((q_c^1 + a, q_c^2 + b), R_\alpha * \text{Rot}(\hat{z}, \theta_1), \text{Rot}(\hat{z}, \theta_2 - \theta_1) * R_\beta) \\ &= ((q_c^1 + a, q_c^2 + b), R_\alpha * \text{Rot}(\hat{z}, \theta_1), R_\beta * \text{Rot}(\hat{z}, \theta_2 - \theta_1)) \\ &= (a, b, R_\alpha, R_\beta)((q_c^1, q_c^2), \text{Rot}(\hat{z}, \theta_1), \text{Rot}(\hat{z}, (\theta_2 - \theta_1))) \\ &= h\bar{h}(\tau). \end{aligned} \quad (21)$$

191 C.3.2 Experimental Details

192 **Datasets:** The water-pouring demonstration trajectories are collected by recording videos of water-
 193 pouring motions of a human demonstrator for 8 seconds (intended for 3.5 seconds of reaching motion
 194 and 4.5 seconds of pouring motion) at 60fps, with three AprilTags, resulting in 480 frames.
 195 Then we extract the SE(3) trajectories of the bottle, whose length $T = 480$. We perform trajectory
 196 smoothing and transform task parameters and the trajectories using group actions of H for the cup's
 197 position to be the origin, and the bottle to be initially in the \hat{x}_s -direction from the bottle, and \hat{x}_b
 198 to be aligned with \hat{x}_s . The resulting task parameters are in the form of $((0, 0), (r, 0, h, R, 0), m_w)$.

199 Assuming the bottle to be initially on the table upright, $r = 0.145$ is constant, and $R = I$. We gather
 200 5 trajectories for 7 different r and 5 different m_w in a total of 175 trajectories. As shown in Figure 3
 201 *Left*, we choose r at every 8cm, from 30cm to 78cm, i.e., $r \in \{0.3, 0.38, 0.46, 0.54, 0.62, 0.7, 0.78\}$,
 202 and as shown in Figure 3 *Right* we choose $m_w \in \{0.2, 0.27, 0.305, 0.34, 0.41\}$. The dimensions of
 203 the cup and the bottle, and the position of the bottle frame are as illustrated in Figure 3 *right*. The
 204 five demonstrations of each task parameter are intended to pour water gradually from the left side of
 205 the cup and the right side of the cup.

206 We use 125 trajectories of $r \in 0.3, 0.38, 0.54, 0.62, 0.78$ as the training dataset, and we randomly
 207 split the other 50 trajectories into half for validation and test dataset. We randomly augment the
 208 dataset 100 and 1,000 times for validation and test dataset respectively, resulting in 2,500 validation
 209 trajectories and 25,000 test trajectories.

210 **Network Architectures and Training Details:** The space of the task parameters is
 211 $\mathbb{R}^2 \times \text{SE}(3) \times [0.2, 0.41]$, where a task parameter can be represented in the form of
 212 $((q_c^1, q_c^2), (q_b^1, q_b^2, q_b^3, R_b), m_w)$. Assuming that the bottle is initially on the table upright, since
 213 q_b^3 is a constant variable and R_b can be represented as $\text{Rot}\hat{z}, \theta_b$, in practical implementation, we use
 214 $(q_c^1, q_c^2, q_b^1, q_b^2, m_w, \cos \theta_b, \sin \theta_b) \in \mathbb{R}^7$ for input of the decoder.

215 The output of the model is an element in $\text{SE}(3)^{480}$ which is not a vector space. A naive paramete-
 216 rization or $\text{SE}(3)$ element (e.g. as a 12-dimensional vector) does not enforce the model outputs to
 217 satisfy $\text{SE}(3)$ constraints. To constraint the model output space to be $\text{SE}(3)^{480}$, we first set all model
 218 output sizes to be $480 \times 6 = 2880$, and add an additional layer Vec2SE3 at the end of every decoder.
 219 Given a vector $v = (v^1, \dots, v^6) \in \mathbb{R}^6$, Vec2SE3 is defined as:

$$\text{Vec2SE3} : v \mapsto \begin{bmatrix} \exp\left(\begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}\right) & \begin{bmatrix} v^4 \\ v^5 \\ v^6 \\ 1 \end{bmatrix} \end{bmatrix} \in \text{SE}(3).$$

220 We finally vectorize the first three rows of the $\text{SE}(3)$ matrix, since the last row is constant at
 221 $(0, 0, 0, 1)$.

222 We use two-layer fully connected neural networks of 168 nodes for the EMMP with elu as its activa-
 223 tion function. TC-VAE’s encoder includes a fully connected network and a temporal convolutional
 224 network, and the decoder includes two fully connected networks for z and τ , a temporal convolutional
 225 network, and a fully connected network. All four fully connected networks used in TC-VAE
 226 are of two layers with size 512. The output sizes of fully connected networks for z and τ in the
 227 decoder are 40 and 80 respectively. The two temporal convolutional layers in TC-VAE are both with
 228 channel sizes $(36, 72, 144)$ and kernel size 3. More details on the structure of TC-VAE are in [26].

229 All models in the experiments have a similar number of parameters, where EMMP contains $(1.51 \times$
 230 $10^6)$ parameters and TC-VAE contains (1.56×10^6) parameters.

231 **Task Parameters for Success Rate Measure:** We sample five feasible trajectories for four task
 232 parameters. Throughout the four task parameters, the cup’s position $(q_c^1, q_c^2) = (-0.2, 0)$, the
 233 bottle is initially in the y -direction from the cup, i.e., $q_b^1 = -0.2$, the bottle is initially aligned
 234 with the base frame, i.e., $R_b = I$. The rest parts, (q_b^2, m_w) for the four task parameters are
 235 $(0.35, 0.25)$, $(0.45, 0.275)$, $(0.40, 0.35)$, $(0.55, 0.400)$. These task parameters are picked within the
 236 robot’s workspace.

237 **Obstacle Avoidance Algorithm:** Given a task parameter τ and an obstacle, the obstacle avoidance
 238 task is performed as follows: (i) we sample z from $p(z)$, (ii) generate the bottle’s trajectories via
 239 $f(z, \tau)$, (iii) check the collision between the bottle and the obstacle and pick collision-free trajec-
 240 tories, and (iv) solve the inverse kinematics problem of the robot and choose one that is feasible and
 241 also collision-free.

242 We check collisions between the bottle and the obstacle and between the robot and the obstacle by
 243 converting the meshes of the bottle and robot to point clouds, and parameterizing the obstacle as
 244 a superquadric, which represents objects as a sign distance function. As a sign distance function,
 245 superquadrics have benefits in checking if a point is inside or outside them. We consider a trajectory
 246 of a point cloud and a superquadric to be collision-free if none of the points in the point cloud gets
 247 inside the superquadric at every timestep, and consider they collide otherwise.

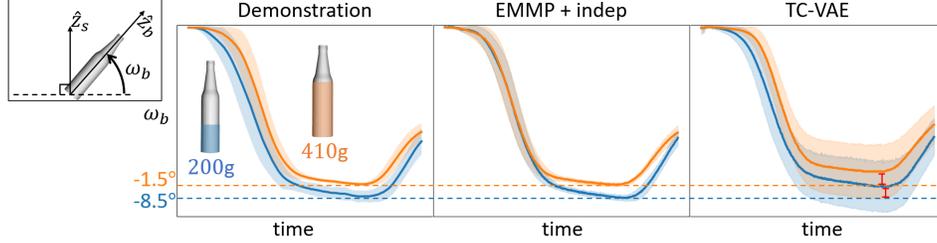


Figure 4: Graphes of bottle angle vs. time. The pouring angle ω_b is the angle between the bottle axis \hat{z}_b and the xy -plane. The orange lines are pouring angles for the $m_w = 0.41$ case, and the blue lines are pouring angles for the $m_w = 0.20$ case. It can be observed that the pouring angle decreases as the mass of water increases.

248 C.3.3 Additional results

249 **Water-Pouring Performance Comparison:** The motions of the bottle pouring water near the cup
 250 are highly dependent on the amount of water in the bottle. A bottle of small water needs to be tilted
 251 more than a bottle that is almost full to pour the same amount of water into the cup. The amount of
 252 tilting of the bottle can be captured in the angle between its axis and the table, which we denote as
 253 the bottle angle.

254 Figure 4 illustrates bottle angle mean and standard deviation graphs of demonstration trajectories
 255 of the training dataset (*Left*), generated trajectories of EMMP + indep (*Middle*) and generated tra-
 256 jectories of TC-VAE (*Right*) with $m_w = 200\text{g}$ (blue) and $m_w = 410\text{g}$ (orange). We randomly
 257 augment 50 task parameters of validation and test datasets 20 times, and pick 1,000 task parameters
 258 for $m_w = 200\text{g}$ and 1,000 task parameters for $m_w = 410\text{g}$. We generate 1,000 trajectories for both
 259 cases using z sampled from $p(z)$.

260 Figure 4 *Left* shows that as the mass of water increases, the pouring angle increases, which means
 261 the bottle is tilted less. It can be seen that the minimum mean angles of EMMP for $m_w = 200\text{g}$
 262 and $m_w = 410\text{g}$ (-1.6 degrees and -9.5 degrees) are very much alike that of the demonstration
 263 trajectories (-1.5 degrees and -8.5 degrees). On the other hand, the minimum mean angles of TC-
 264 VAE (5.6 degrees and -3.1 degrees) are very much distant from the demonstration trajectories’.

265 **Equivariance Comparison:** For a motion manifold primitive framework to be equivariant, decoded
 266 trajectories must equivariantly transform as task parameters undergo a symmetry transformation.
 267 We qualitatively compare the equivariance performance of random data augmentation method and
 268 equivariant learning method by comparing TC-VAE and EMMP + indep.

269 Figure 5 *Left* visualizes two trajectories generated from τ and $h \cdot \tau$, where h is the rotation of the
 270 bottle around the cup and itself, without translation. If the model is equivariant, the orange-colored
 271 bottle and the apricot-colored bottle in the left upper corner should overlap. However, the condition
 272 is not satisfied for TC-VAE, whereas the orange trajectory of EMMP is equivariantly transformed
 273 with τ .

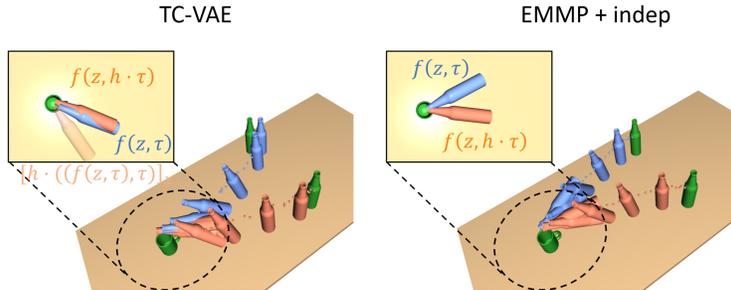


Figure 5: $f(z, \tau)$ (blue) and $f(z, h \cdot \tau)$ (orange), and $[h \cdot (f(z, \tau), \tau)]_x$ (apricot). $[h \cdot (f(z, \tau), \tau)]_x$ and $f(z, h \cdot \tau)$ should be overlapped if the trajectories are generated equivariantly with the task parameters.

274 **References**

- 275 [1] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel. Dynamic movement primi-
276 tives in robotics: A tutorial survey. *arXiv preprint arXiv:2102.03861*, 2021.
- 277 [2] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement
278 primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–
279 373, 2013.
- 280 [3] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Trajectory formation for imitation with nonlinear
281 dynamical systems. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent*
282 *Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat.*
283 *No. 01CH37180)*, volume 2, pages 752–757. IEEE, 2001.
- 284 [4] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration
285 using nonlinear oscillators. In *Proceedings of the ieee/rsj int. conference on intelligent robots*
286 *and systems (iros2002)*, number CONF, pages 958–963, 2002.
- 287 [5] S. Schaal, P. Mohajerian, and A. Ijspeert. Dynamics systems vs. optimal control—a unifying
288 view. *Progress in brain research*, 165:425–445, 2007.
- 289 [6] A. Pervez, A. Ali, J.-H. Ryu, and D. Lee. Novel learning from demonstration approach for
290 repetitive teleoperation tasks. In *2017 IEEE World Haptics Conference (WHC)*, pages 60–65.
291 IEEE, 2017.
- 292 [7] Y. Fanger, J. Umlauf, and S. Hirche. Gaussian processes for dynamic movement primitives
293 with application in knowledge-based cooperation. In *2016 IEEE/RSJ International Conference*
294 *on Intelligent Robots and Systems (IROS)*, pages 3913–3919. IEEE, 2016.
- 295 [8] J. Umlauf, Y. Fanger, and S. Hirche. Bayesian uncertainty modeling for programming by
296 demonstration. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*,
297 pages 6428–6434. IEEE, 2017.
- 298 [9] A. Pervez, Y. Mao, and D. Lee. Learning deep movement primitives using convolutional
299 neural networks. In *2017 IEEE-RAS 17th international conference on humanoid robotics (Hu-*
300 *manoids)*, pages 191–197. IEEE, 2017.
- 301 [10] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak. Neural dynamic policies for end-to-end
302 sensorimotor learning. *Advances in Neural Information Processing Systems*, 33:5058–5069,
303 2020.
- 304 [11] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian
305 mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.
- 306 [12] K. Neumann, A. Lemme, and J. J. Steil. Neural learning of stable dynamical systems based on
307 data-driven lyapunov candidates. In *2013 IEEE/RSJ International Conference on Intelligent*
308 *Robots and Systems*, pages 1216–1222. IEEE, 2013.
- 309 [13] S. M. Khansari-Zadeh and A. Billard. Learning control lyapunov function to ensure stability
310 of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):
311 752–765, 2014.
- 312 [14] A. Lemme, K. Neumann, R. F. Reinhart, and J. J. Steil. Neural learning of vector fields for
313 encoding stable dynamical systems. *Neurocomputing*, 141:3–14, 2014.
- 314 [15] K. Neumann and J. J. Steil. Learning robot motions with stable dynamical systems under
315 diffeomorphic transformations. *Robotics and Autonomous Systems*, 70:1–15, 2015.
- 316 [16] C. Blocher, M. Saveriano, and D. Lee. Learning stable dynamical systems using contraction
317 theory. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence*
318 *(URAI)*, pages 124–129. IEEE, 2017.
- 319 [17] N. Figueroa and A. Billard. A physically-consistent bayesian non-parametric mixture model
320 for dynamical system learning. In *CoRL*, pages 927–946, 2018.

- 321 [18] V. Sindhvani, S. Tu, and M. Khansari. Learning contracting vector fields for stable imitation
322 learning. *arXiv preprint arXiv:1804.04878*, 2018.
- 323 [19] J. Z. Kolter and G. Manek. Learning stable deep dynamics models. *Advances in neural infor-*
324 *mation processing systems*, 32, 2019.
- 325 [20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint*
326 *arXiv:1312.6114*, 2013.
- 327 [21] Y. Lee, H. Kwon, and F. Park. Neighborhood reconstructing autoencoders. *Advances in Neural*
328 *Information Processing Systems*, 34:536–546, 2021.
- 329 [22] Y. Lee, S. Yoon, M. Son, and F. C. Park. Regularized autoencoders for isometric representation
330 learning. In *International Conference on Learning Representations*, 2022.
- 331 [23] C. Jang, Y. Lee, Y.-K. Noh, and F. C. Park. Geometrically regularized autoencoders for non-
332 euclidean data. In *The Eleventh International Conference on Learning Representations*.
- 333 [24] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit
334 invariance during feature extraction. In *Proceedings of the 28th international conference on*
335 *international conference on machine learning*, pages 833–840, 2011.
- 336 [25] A. Creswell, Y. Mohamied, B. Sengupta, and A. A. Bharath. Adversarial information factor-
337 ization. *arXiv preprint arXiv:1711.05175*, 2017.
- 338 [26] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy. Task-conditioned variational autoen-
339 coders for learning movement primitives. In *Conference on robot learning*, pages 933–944.
340 PMLR, 2020.
- 341 [27] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids,
342 groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.
- 343 [28] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural net-
344 work. In *2017 international conference on engineering and technology (ICET)*, pages 1–6.
345 Ieee, 2017.
- 346 [29] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International confer-*
347 *ence on machine learning*, pages 2990–2999. PMLR, 2016.
- 348 [30] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. *arXiv preprint*
349 *arXiv:1801.10130*, 2018.
- 350 [31] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitz-
351 mann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In
352 *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE,
353 2022.
- 354 [32] S. Kim, B. Lim, Y. Lee, and F. C. Park. Se (2)-equivariant pushing dynamics models for
355 tabletop object manipulations. In *Conference on Robot Learning*, pages 427–436. PMLR,
356 2023.
- 357 [33] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm.
358 Mutual information neural estimation. In *International conference on machine learning*, pages
359 531–540. PMLR, 2018.