

Exploiting Symmetry in Dynamics for Model-Based Reinforcement Learning with Asymmetric Rewards

Yasin Sonmez, Neelay Junnarkar, Murat Arcak

Abstract—Recent work in reinforcement learning has leveraged symmetries in the model to improve sample efficiency in training a policy. A commonly used simplifying assumption is that the dynamics and reward both exhibit the same symmetry. However, in many real-world environments, the dynamical model exhibits symmetry independent of the reward model: the reward may not satisfy the same symmetries as the dynamics. In this paper, we investigate scenarios where only the dynamics are assumed to exhibit symmetry, extending the scope of problems in reinforcement learning and learning in control theory where symmetry techniques can be applied. We use Cartan’s moving frame method to introduce a technique for learning dynamics which, by construction, exhibit specified symmetries. We demonstrate through numerical experiments that the proposed method learns a more accurate dynamical model.

I. INTRODUCTION

A significant area of research in reinforcement learning (RL) is in improving sample efficiency, the amount a learning method must interact with the environment to learn a good policy. While model-free methods such as [1], [2], and [3] have shown the ability to achieve high rewards in many complex environments, they typically require many environment interactions. Model-based reinforcement learning (MBRL) methods, on the other hand, have shown promise in learning policies quickly relative to the number of environment interactions. This improvement in sample efficiency confirms that models contain a great deal of information useful for improving policies [4].

One idea gaining recent attention in RL methods is to integrate symmetries into model-based information. Using knowledge of symmetries, one can extrapolate observed behavior to other scenarios. Recent work has leveraged symmetries via equivariant networks [5], [6], which enforce certain transformation relationships on the inputs and outputs, and via data augmentation [7], where artificial data is created based on observed data and known transformations for training. However, a limitation of equivariant networks is that they must be designed for the particular group of symmetries to which they are equivariant. Unlike the above methods, which assume that the symmetries are known apriori, reference [7] learns symmetries from data and augments the training data using these symmetries.

This work was supported in part by the National Science Foundation award CNS-2111688 and by the Air Force Office of Scientific Research grant FA9550-21-1-0288.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94720 USA (emails: yasin.sonmez@berkeley.edu, neelay.junnarkar@berkeley.edu, arcak@berkeley.edu).

Methods that leverage symmetry have also been explored for control design and verification. Reference [8] uses symmetry to deduce the structure of the optimal controllers for nonlinear systems. Reference [9] uses symmetry to improve the convergence of observers. Reference [10] explores the use of symmetry for reducing the dimension of variables involved in dynamic programming, accelerating the computation of control policies. Reference [11] uses symmetry reduction to accelerate reachability computations. References [12] and [13] use symmetry in accelerating safety verification.

A common assumption in the use of symmetry in both RL and optimal control is that both the dynamics and reward (or cost) function exhibit the same symmetries. This combination of assumptions results in the optimal policy (or control law) likewise being symmetric [10], [14]. The symmetric policy structure is exploited for Markov Decision Processes (MDPs) and RL in [5], [6], [14], [15], and for dynamic programming in [10].

In many scenarios, however, symmetries in the dynamical model are not necessarily applicable to the reward model. For example, multiple tasks might be executed in the same environment, where the same dynamical symmetries exist, while the reward model changes from task to task. Exploiting symmetries only in the dynamical model widens the scope of RL problems to which symmetry can be applied.

Motivated by commonly encountered translation and rotation symmetries in dynamics, we consider a method of enforcing continuous symmetries through transformation into a reduced coordinate space. For example, the dynamics of the cart-pole system, commonly used for control systems pedagogy, are invariant to the position of the cart. This fact can be exploited when learning the system dynamics by simply removing the position data from the training dataset. It is, however, less clear how to account for symmetries that involve more complex transformations.

In this paper, we present a method to learn dynamical models that are, by construction, invariant under specified continuous symmetries. This enables encoding apriori known symmetry structure when learning a dynamical model. To address a large class of symmetries, we use Cartan’s moving frame method [16], which simplifies models by adapting moving reference frames to the structure of the space.

This paper is organized as follows: in Section II, we provide background information on symmetries, and then use Cartan’s method of moving frames to relate a symmetrical dynamical model to a function operating on an input space of reduced dimension. In Section III, we use the result of Section II to learn a dynamical model which, by construction,

satisfies specified symmetries. In Section IV we apply symmetry reduction to learn dynamical models for two examples having symmetry in dynamics.

II. SYMMETRIES IN MODEL DYNAMICS

Consider the deterministic discrete-time system

$$x_{k+1} = F(x_k, u_k) \quad (1)$$

where $x_k \in \mathcal{X} = \mathbb{R}^n$ is the state, $u_k \in \mathcal{U} = \mathbb{R}^{n_u}$ is the control input, and $k \in \mathbb{N}$ denotes the time step.

Following [9] and [11], we provide definitions of a transformation group and of dynamics invariant under a transformation group.

Definition 1 (Transformation Group): Let G be a group. A transformation group on $\mathcal{X} \times \mathcal{U}$ is a set of tuples $(\phi_\alpha, \psi_\alpha)$ parameterized by $\alpha \in G$ such that $\phi_\alpha : \mathcal{X} \rightarrow \mathcal{X}$ and $\psi_\alpha : \mathcal{U} \rightarrow \mathcal{U}$ are diffeomorphisms that satisfy the following criteria for all $x \in \mathcal{X}$, $u \in \mathcal{U}$, and $g_1, g_2 \in G$:

- $\phi_e(x) = x$ and $\psi_e(u) = u$ where e is the identity of group G and
- $\phi_{g_1}(\phi_{g_2}(x)) = \phi_{g_1 \cdot g_2}(x)$ and $\psi_{g_1}(\psi_{g_2}(u)) = \psi_{g_1 \cdot g_2}(u)$ where $g_1 \cdot g_2$ denotes the group operation. We will denote this simply by $g_1 g_2$ henceforth.

Definition 2: We say the system (1) is *invariant* to the transformation group $\{(\phi_\alpha, \psi_\alpha)\}_{\alpha \in G}$, or G -invariant, if

$$F(\phi_\alpha(x), \psi_\alpha(u)) = \phi_\alpha(F(x, u)) \quad (2)$$

for all $x \in \mathcal{X}$, $u \in \mathcal{U}$, and $\alpha \in G$.

We now use Cartan's Moving Frame method [16] to parameterize the system (1) in terms of a function with lower dimensional input space, when (1) is invariant to a transformation group parameterized by a Lie group.

Let G be a Lie group which acts on the state space with diffeomorphisms $\{\phi_\alpha\}_{\alpha \in G}$, and acts on the control input space with diffeomorphisms $\{\psi_\alpha\}_{\alpha \in G}$. Assume that G is r -dimensional, with $r \leq n$. Now, split ϕ_α into $(\phi_\alpha^a, \phi_\alpha^b)$ with r and $n-r$ components respectively, such that ϕ_α^a is invertible with respect to α . Pick c that is in the range of ϕ^a , and define $\mathcal{C} = \{x | \phi_e^a(x) = c\}$ where $e \in G$ is the group identity. Note that $\mathcal{C}$ is $n-r$ dimensional. This is the lower dimensional space that will be used to parameterize system (1).

Assume that, for all x , there exists a unique $\alpha \in G$ such that $\phi_\alpha(x) \in \mathcal{C}$. Then, we define a function γ such that $\phi_{\gamma(x)}(x) \in \mathcal{C}$. Using the function γ , which is called a *moving frame*, we define the following map, $\rho : \mathcal{X} \rightarrow \mathcal{X}^b$:

$$\rho(x) \triangleq \phi_{\gamma(x)}^b(x). \quad (3)$$

This ρ is invariant to the action of G on the state, as shown in the following lemma adapted from [9].

Lemma 1: For all $\alpha \in G$ and $x \in \mathcal{X}$, $\rho(\phi_\alpha(x)) = \rho(x)$.

Proof: Let $\alpha \in G$. Then,

$$\begin{aligned} \rho(\phi_\alpha(x)) &= \phi_{\gamma(\phi_\alpha(x))}^b(\phi_\alpha(x)) \\ &= \phi_{\gamma(\phi_\alpha(x))\alpha}^b(x). \end{aligned}$$

From the definition of γ , we have that $\phi_{\gamma(x)}^a(x) = c$. Note that $\phi_{\gamma(\phi_\alpha(x))}^a(\phi_\alpha(x)) = \phi_{\gamma(\phi_\alpha(x))\alpha}^a(x)$. Therefore,

$\gamma(\phi_\alpha(x))\alpha = \gamma(x)$. Plugging this into the expression for $\rho(\phi_\alpha(x))$ gives $\rho(\phi_\alpha(x)) = \phi_{\gamma(x)}^b(x) = \rho(x)$. ■

The map ρ outputs the element of $\mathcal{C}$, eliminating the part equal to c , that is related by a symmetry transformation to the state x . This effectively transforms the state space from dimension n to dimension $n-r$. The following theorem shows the dynamics can be evaluated using the lower dimensional output of ρ .

Theorem 1: The system (1), $F : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$, is G -invariant if and only if there exists a map $\bar{F} : \mathcal{X}^b \times \mathcal{U} \rightarrow \mathcal{X}$ such that

$$\phi_{\gamma(x)}(F(x, u)) = \bar{F}(\rho(x), \psi_{\gamma(x)}(u)) \quad (4)$$

for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$.

Proof: Assume that the dynamics F are G -invariant. Note that ρ restricted to $\mathcal{C}$ is invertible. Define $\bar{F}$ by $\bar{F}(\bar{x}, u) = F(\rho|_{\mathcal{C}}^{-1}(\bar{x}), u)$. Letting $x = \rho|_{\mathcal{C}}^{-1}(\bar{x})$, this can equivalently be written as $\bar{F}(\rho(x), u) = F(x, u)$ when $x \in \mathcal{C}$. Since F is G -invariant, $\phi_{\gamma(x)}(F(x, u)) = F(\phi_{\gamma(x)}(x), \psi_{\gamma(x)}(u))$. Note that $\phi_{\gamma(x)}(x) \in \mathcal{C}$. This can be rewritten in terms of $\bar{F}$ as $\bar{F}(\rho(\phi_{\gamma(x)}(x)), \psi_{\gamma(x)}(u))$. Since ρ is invariant to action of G , this equals $\bar{F}(\rho(x), \psi_{\gamma(x)}(u))$, as desired.

Now we prove the reverse direction. Assume there exists $\bar{F} : \mathcal{X}^b \times \mathcal{U} \rightarrow \mathcal{X}$ such that (4) for all $x \in \mathcal{X}$, $u \in \mathcal{U}$. Let $\alpha \in G$. Then,

$$\begin{aligned} F(\phi_\alpha(x), \psi_\alpha(u)) &= \phi_{\gamma(\phi_\alpha(x))}^{-1}(\bar{F}(\rho(\phi_\alpha(x)), \psi_{\gamma(\phi_\alpha(x))}(\psi_\alpha(u)))) \\ &= \phi_{\gamma(\phi_\alpha(x))}^{-1}(\bar{F}(\rho(x), \psi_{\gamma(\phi_\alpha(x))\alpha}(u))). \end{aligned}$$

From the proof of Lemma 1, we know $\gamma(\phi_\alpha(x))\alpha = \gamma(x)$. Therefore the above can be simplified to

$$\begin{aligned} F(\phi_\alpha(x), \psi_\alpha(u)) &= \phi_{\gamma(x)\alpha}^{-1}(\bar{F}(\rho(x), \psi_{\gamma(x)}(u))) \\ &= \phi_\alpha(\phi_{\gamma(x)}^{-1}(\phi_{\gamma(x)}(F(x, u)))) \\ &= \phi_\alpha(F(x, u)). \end{aligned}$$

Thus, F is G -invariant. ■

III. LEARNING MODEL DYNAMICS

A critical part of the theory in Section II is that it depends only on the state space, control input space, and the symmetries of the system; we do not require knowledge of the system dynamics.

We learn a dynamical model that is invariant to a given transformation group with the help of Theorem 1. Unlike a standard formulation for learning a dynamical model, which would learn $F : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$, we instead learn the function $\bar{F} : \mathcal{X}^b \times \mathcal{U} \rightarrow \mathcal{X}$ from the statement of Theorem 1. With this function $\bar{F}$, we construct the model dynamics F as

$$F(x, u) \triangleq \phi_{\gamma(x)}^{-1}(\bar{F}(\rho(x), \psi_{\gamma(x)}(u))). \quad (5)$$

The relationship between F and $\bar{F}$ is depicted in Figure 1. From Theorem 1, F is, by construction, G -invariant. A side benefit is that training is done with a lower dimensional input

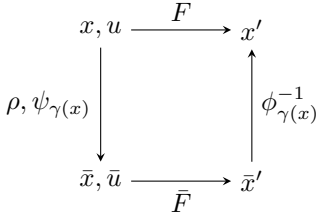


Fig. 1. Relationship between F and $\bar{F}$.

space $(\mathcal{X}^b \times \mathcal{U}$ instead of $\mathcal{X} \times \mathcal{U}$). To learn this function $\bar{F}$, given a dataset $\mathcal{D}$ consisting of tuples (x_k, u_k, x_{k+1}) , we train $\bar{F}$ to map $(\rho(x_k), \psi_{\gamma(x_k)}(u_k))$ to $\phi_{\gamma(x_k)}(x_{k+1})$.

Note in practice it is typical to learn a function which maps (x_k, u_k) to $x_{k+1} - x_k$ instead of x_{k+1} directly. To do this, we learn $\Delta\bar{F}$ to map $(\rho(x_k), \psi_{\gamma(x_k)}(u_k))$ to $\phi_{\gamma(x_k)}(x_{k+1}) - \phi_{\gamma(x_k)}(x_k)$. Then, we construct F by

$$F(x, u) = \phi_{\gamma(x)}^{-1}(\Delta\bar{F}(\rho(x), \psi_{\gamma(x)}(u)) + \phi_{\gamma(x)}(x)).$$

This construction for F is G -invariant since it is equivalent to using $\bar{F}(\bar{x}, \bar{u}) \triangleq \Delta\bar{F}(\bar{x}, \bar{u}) + \rho|_{\mathcal{C}}^{-1}(\bar{x})$, which satisfies the condition in Theorem 1. Finally, $\Delta F(x, u)$ can be defined as $\Delta F(x, u) \triangleq F(x, u) - x$ to provide the same $x_{k+1} - x_k$ interface expected by many libraries. This construction for ΔF in terms of $\Delta\bar{F}$ expands as follows.

$$\Delta F(x, u) \triangleq \phi_{\gamma(x)}^{-1}(\Delta\bar{F}(\rho(x), \psi_{\gamma(x)}(u)) + \phi_{\gamma(x)}(x)) - x \quad (6)$$

When ϕ_α is a homomorphism with respect to $+$ for all $\alpha \in G$, (6) simplifies to

$$\Delta F(x, u) = \phi_{\gamma(x)}^{-1}(\Delta\bar{F}(\rho(x), \psi_{\gamma(x)}(u))).$$

An example where this occurs is when ϕ_α is a linear transformation.

IV. EXPERIMENTS

In this section, we demonstrate the performance of this method on two numerical examples involving learning the dynamics of a system. Two environments that we used were “Parking” from “Highway-env” [17] and “Reacher” from OpenAI Gym [18] exhibiting rotation and/or translation symmetry which are explained in Figure 2. In “Parking” there are two controlled vehicles that we need to park to corresponding parking spots and in “Reacher” we control a robot arm with 2 joints to reach to a target position. To be able to compare our method fairly to a baseline we generated a static dataset of transitions (x, u, x') , and we learn a dynamical model from this dataset. Consequently, the comparison between the two methodologies was centered on their respective capabilities in dynamics learning rather than policy generation. This approach ensures a fair comparison, as it relies on the same offline dataset for both methods, avoiding discrepancies that may arise from online training and dataset variations.

The dataset is constructed by training an agent using Soft Actor-Critic [2], a model-free reinforcement learning method. We use D3rlpy [19] to train dynamical models

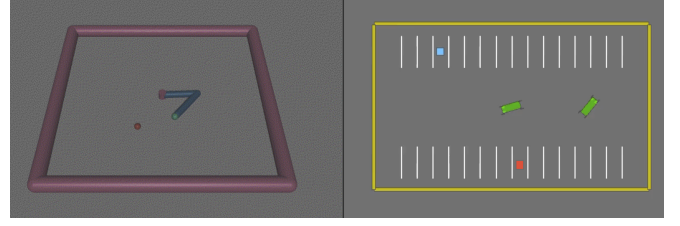


Fig. 2. Experimental environments included: (1) “Reacher” on the left featuring two rotating controlled joints that exhibit rotation symmetry with the objective of reaching a target point, and (2) “Parking” involving two controlled vehicles maneuvering to park in designated spots without collision, demonstrating both rotational and translational symmetry in dynamics.

from the pre-collected dataset. We implement the symmetry method discussed above using (5), where $\bar{F}$ is the neural network whose parameters are trained. The observation error (on the test dataset) of this dynamical model is compared against the observation error of learning F directly. We further compare these methods (with and without symmetry) across a variety of parameter sizes. The rollouts from the resulting policy, the datasets used in this work and the code to reproduce the results in this paper can be found on our GitHub repository¹.

A. Two Cars Parking Scenario

We use the parking environment from [17] with two cars with separate goal positions. The aim of this environment is to park the two cars at their goal parking spots by controlling them separately and not allowing them to crash into obstacles or to each other.

Note that the dynamics of each car are translation- and rotation-invariant, but the reward function (relating to the location and orientation of the parking spot) does not follow the same symmetry. Therefore it is not possible to employ methods that assume that both the dynamics and the reward function are symmetric. The car dynamics in this environment follow the kinematic bicycle model [20], with state $x = (y, z, v_y, v_z, h_y, h_z)$ corresponding to the y position, z position, y component of velocity, z component of velocity, cosine of heading angle, and sine of heading angle.

The state of this parking environment is 24-dimensional, with the first 6 states corresponding to the first car, the second 6 states corresponding to the second car, the third 6 states corresponding to the first car’s goal, and the final 6 states corresponding to the second car’s goal. This state corresponds to 4 independent systems: the two cars follow the kinematic bicycle model, and the two goal systems are static, following the simple dynamical model of $g_{k+1} = g_k$. We apply the method presented in Sections II and III to construct transformation groups for each of the four systems. The joint system then satisfies the transformation group formed as the product of the individual transformation groups.

We first describe the transformation group applied to learn translation- and rotation-invariant dynamics for the two car

¹<https://github.com/YasinSonmez/symmetry-cs285>

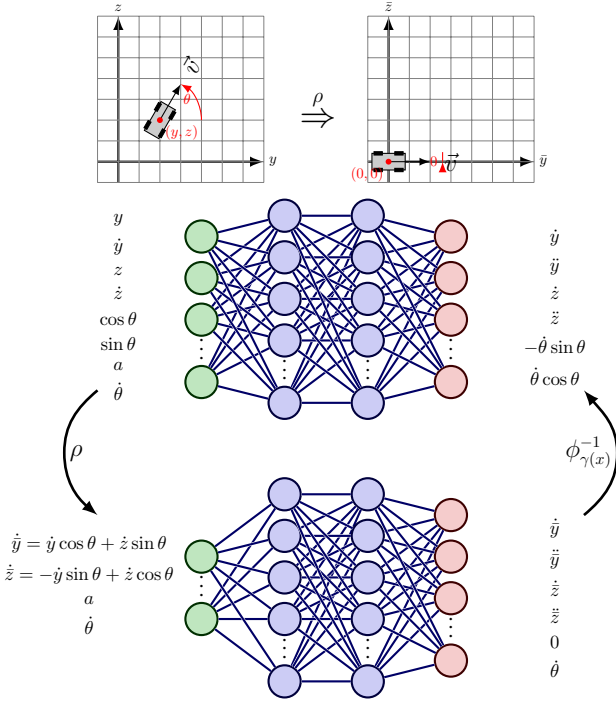


Fig. 3. Illustration of car dynamics demonstrating translational and rotational invariance. By applying Cartan’s moving frame method, coordinates are transformed to position the car at the origin with a neutral orientation. The function ρ reduces the system’s state to a lower-dimensional space, without losing essential dynamics information. Subsequently, dynamics are learned in these modified coordinates via a smaller neural network (bottom) compared to the usual NN (middle). The NN’s output is then reconverted to original coordinates using $\phi_{\gamma(x)}^{-1}$.

models. We parameterize the symmetry group $G = SE(2)$ by a translation (y', z') and a rotation θ' . Define the rotation matrix

$$R_{\theta'} = \begin{bmatrix} \cos \theta' & -\sin \theta' \\ \sin \theta' & \cos \theta' \end{bmatrix}. \quad (7)$$

To define the action of G on the state space, let $(y', z', \theta') \in G$ and x be the 6-dimensional state of the car:

$$\phi_{(y', z', \theta')}(x) = \begin{bmatrix} R_{\theta'} & 0 & 0 \\ 0 & R_{\theta'} & 0 \\ 0 & 0 & R_{\theta'} \end{bmatrix} x + \begin{bmatrix} y' \\ z' \\ 0 \end{bmatrix} \quad (8)$$

In this case, the action of G on the control input is the identity: $\psi_{\alpha}(u) = u$ for all $\alpha \in G$.

We now derive $\gamma(x)$, $\gamma(x)^{-1}$ (group inverse), and ρ for this transformation group, so we can evaluate (5). Define $\mathcal{C}$ to be the set of states whose position is equal to 0 ($y = z = 0$) and whose heading is equal to 0 ($h_y = 1$ and $h_z = 0$). The moving frame γ is then defined as the solution to:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \phi_{\gamma(x)}^a(x) = \begin{bmatrix} R_{\theta'} & 0 \\ 0 & R_{\theta'} \end{bmatrix} \begin{bmatrix} y \\ z \\ h_y \\ h_z \end{bmatrix} + \begin{bmatrix} y' \\ z' \\ 0 \\ 0 \end{bmatrix},$$

which is given by:

$$\gamma(x) = \begin{bmatrix} -yh_y - zh_z \\ yh_z - zh_y \\ \arctan2(-h_z, h_y) \end{bmatrix}. \quad (9)$$

We can then compute the group inverse of $\gamma(x)$ as:

$$\gamma(x)^{-1} = \begin{bmatrix} y \\ z \\ \arctan2(h_z, h_y) \end{bmatrix}. \quad (10)$$

Substituting $\gamma(x)$ into the definition of ρ , we get:

$$\rho(x) = \begin{bmatrix} h_y v_y + h_z v_z \\ -h_z v_y + h_y v_z \end{bmatrix}. \quad (11)$$

Now we present the transformation group applied to the goal dynamical model. The goal dynamical model satisfies the trivial transformation group parameterized by $(\mathbb{R}^6, +)$ such that $\phi_{\delta}(g) = g + \delta$, and there is no input. The “a” component of ϕ has 6 components, leaving the “b” component with 0. Effectively, we remove the goal states entirely.

Using the above transformation groups applied to each car and goal, the transformation group applied to the joint state of the environment gives a ρ which reduces the state size from 24 to 4. Thus, we learn a neural network with significantly reduced input size. Note that the computation of these functions only depends on the symmetry and not on the dynamics, so the dynamical model need not be known apriori and can be learned using the data. This procedure is explained in Figure 3.

The three neural network configurations that we used in our experiments are: 1 hidden layer with size 128, 2 hidden layers with size 128, and 3 hidden layers with size 128. In the method using symmetry, the input size is 8 (reduced coordinate size for each car of 2 and control input size of 2 for each of the two cars), and the output size is 24. In the method not using symmetry, the input size is 28 (state space size of 24 and control input size of 2 for each of the two cars), and the output size is 24.

The results of observation error vs. number of parameter updates are shown in Figure 4. At the lower number of parameters, the dynamical model training with symmetry outperforms the default method without symmetry by achieving a lower observation error. When there are two hidden layers, the method without symmetry learns slower than does the method with symmetry, but ultimately catches up in performance. When there are three hidden layers, the method without symmetry exceeds the performance of that of the method with symmetry.

B. Reacher

In this experiment, we consider the standard reacher environment in OpenAI Gym. The dynamics of the reacher exhibit rotational symmetry with the first joint angle, but the target does not since it is fixed. We use symmetry reduction to learn model dynamics which are rotation-invariant.

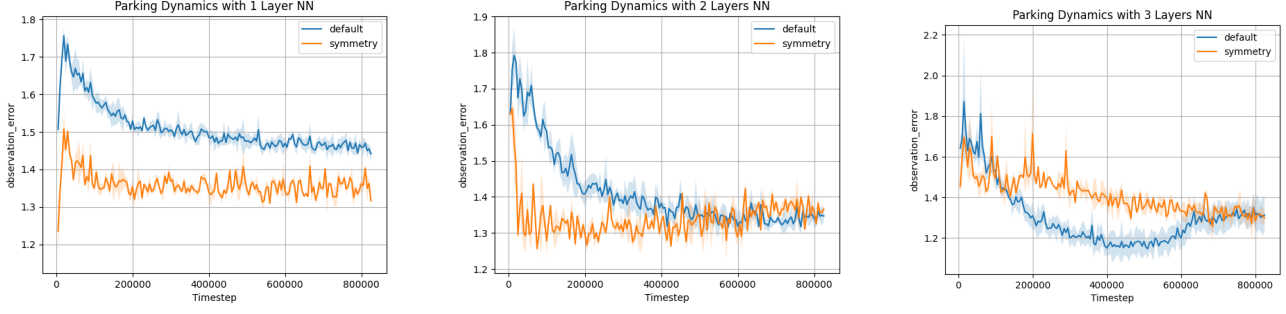


Fig. 4. Comparison of learning the dynamics with and without using symmetry in parking scenario with different NN architectures. The y-axis (observation error) is the error on the test dataset. Mean and standard deviation reported over 4 runs.

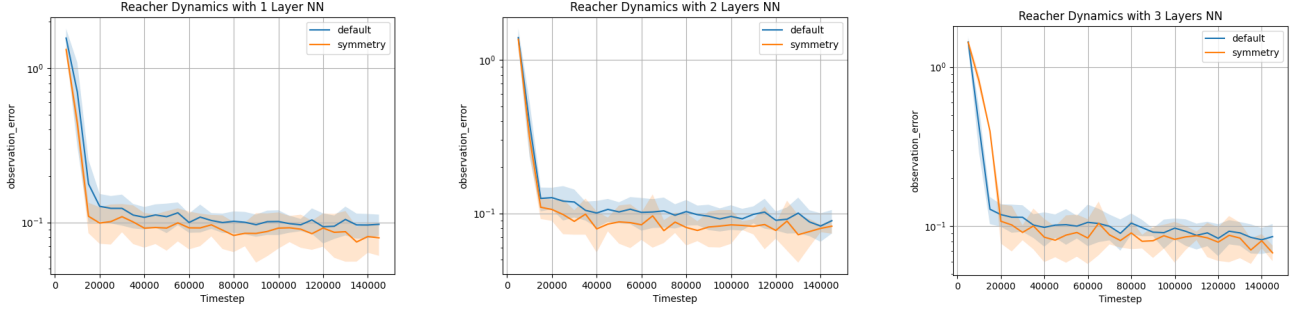


Fig. 5. Comparison of learning the dynamics with and without using symmetry in reacher environment with different NN architectures. The y-axis (observation error) is the error on the test dataset. Mean and standard deviation reported over 4 runs.

The state of the reacher is 11-dimensional, with x_1 and x_2 representing the cosines of the first and second joint angles, x_3 and x_4 representing the sines of the joint angles, x_5 and x_6 representing the x- and y-coordinates of the target, x_7 and x_8 representing the angular velocities of the first and second arm, x_9 and x_{10} representing the distance between the reacher fingertip and the target along the x- and y-axes, and x_{11} equaling the constant 0 representing the z coordinate of the fingertip.

We now describe the transformation group applied to learn a rotation-invariant dynamical model for the reacher. For x_5, x_6 , and x_{11} , we apply the same symmetry used in Section IV-A to describe constants. Let G be parameterized by $\alpha = (\theta', \delta_1, \delta_2, \delta_3) \in \mathbb{R}^4$, where θ' represents the angle by which the reacher is rotated, and δ represents translations of the target position and x_{11} . We define the action of G on the state space as:

$$\phi_\alpha(x) = \begin{bmatrix} \cos(\theta')x_1 - \sin(\theta')x_3 \\ x_2 \\ \sin(\theta')x_1 + \cos(\theta')x_3 \\ x_4 \\ \cos(\theta')(x_5 + \delta_1) - \sin(\theta')(x_6 + \delta_2) \\ \sin(\theta')(x_5 + \delta_1) + \cos(\theta')(x_6 + \delta_2) \\ x_7 \\ x_8 \\ \cos(\theta')(x_9 - \delta_1) - \sin(\theta')(x_{10} - \delta_2) \\ \sin(\theta')(x_9 - \delta_1) + \cos(\theta')(x_{10} - \delta_2) \\ x_{11} + \delta_3 \end{bmatrix}. \quad (12)$$

The action of G on the control input is the identity: $\psi_\alpha(u) = u$ for all $\alpha \in G$.

To define the symmetry reduction, we define $\mathcal{C}$ by the states x for which $x_1 = 1$, $x_3 = 0$, $x_5 = 0$, $x_6 = 0$, and $x_{11} = 0$. The solution for the moving frame $\gamma(x)$ is

$$\gamma(x) = \begin{bmatrix} \arctan2(-x_3, x_1) \\ -x_5 \\ -x_6 \\ -x_{11} \end{bmatrix},$$

and the solution for the group inverse of $\gamma(x)$ is

$$\gamma(x)^{-1} = \begin{bmatrix} \arctan2(x_3, x_1) \\ x_1x_5 + x_3x_6 \\ -x_3x_5 + x_1x_6 \\ x_{11} \end{bmatrix}.$$

We now plug γ into (3) to solve for ρ :

$$\rho(x) = \begin{bmatrix} x_2 \\ x_4 \\ x_7 \\ x_8 \\ x_1(x_9 + x_5) + x_3(x_{10} + x_6) \\ -x_3(x_9 + x_5) + x_1(x_{10} + x_6) \end{bmatrix}. \quad (13)$$

Using this symmetry group, the input size of the neural network is 8 (reduced coordinate size 6 and control input size of 2), and the output size is 11. In the method not using symmetry, the input size is 13 (state space size of 11 and control input size of 2), and the output size is 11.

The three neural network configurations that we used in our experiments are: 1 hidden layer with size 64, 2 hidden layers with size 64, and 3 hidden layers with size 64. We used 64 neuron size compared to 128 in “Parking” because of the simplicity of this problem.

The results of observation error vs. number of parameter updates are shown in Figure 5. In all cases, the dynamical model training with symmetry achieves slightly better performance than the default method without symmetry by achieving a lower observation error at the end of the training. This is most evident when the neural network has the least number of parameters, in the 1-layer case.

V. CONCLUSION

This study highlights the benefits of exploiting symmetry when learning dynamical models. By focusing on the symmetry inherent in dynamics, independent of the reward function, we broaden the range of problems where symmetry can be effectively utilized. We investigated symmetries such as rotation and translation symmetries through Cartan’s method of moving frames and our proposed method proves to be effective in leveraging these symmetries to learn more accurate models, particularly at low numbers of model parameters. The experiments confirm the potential of our approach to enhance learning efficiency in MBRL settings. We compare our method with varying parameter sizes and see a higher benefit of utilizing symmetry in smaller networks.

VI. ACKNOWLEDGEMENTS

We thank Sergey Levine for pointing to relevant references and for discussions on the preliminary results. Additionally, we thank Hussein Sibai for discussions on symmetry reduction that eventually led to this formulation of the paper.

REFERENCES

- [1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, *Continuous control with deep reinforcement learning*, 2019. arXiv: 1509.02971 [cs.LG].
- [2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proceedings of the 35th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 3, 2018, pp. 1861–1870. [Online]. Available: <https://proceedings.mlr.press/v80/haarnoja18b.html>.
- [3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].
- [4] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, “Model-based reinforcement learning: A survey,” *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023, ISSN: 1935-8237. DOI: 10.1561/22000000086. [Online]. Available: <http://dx.doi.org/10.1561/22000000086>.
- [5] E. van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling, “MDP homomorphic networks: Group symmetries in reinforcement learning,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 4199–4210. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/2be5f9c2e3620eb73c2972d7552b6cb5-Paper.pdf.
- [6] D. Wang, R. Walters, and R. Platt, “SO(2)-equivariant reinforcement learning,” in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=7F9c0hdvfk_.
- [7] M. Weissenbacher, S. Sinha, A. Garg, and K. Yoshinobu, “Koopman Q-learning: Offline reinforcement learning via symmetries of dynamics,” in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., ser. Proceedings of Machine Learning Research, vol. 162, PMLR, Jul. 2022, pp. 23 645–23 667. [Online]. Available: <https://proceedings.mlr.press/v162/weissenbacher22a.html>.
- [8] J. Grizzle and S. Marcus, “Optimal control of systems possessing symmetries,” *IEEE Transactions on Automatic Control*, vol. 29, no. 11, pp. 1037–1040, Nov. 1984, ISSN: 1558-2523. DOI: 10.1109/TAC.1984.1103421. [Online]. Available: <https://ieeexplore.ieee.org/document/1103421>.
- [9] S. Bonnabel, P. Martin, and P. Rouchon, “Symmetry-preserving observers,” *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2514–2526, 2008. DOI: 10.1109/TAC.2008.2006929.
- [10] J. Maidens, A. Barrau, S. Bonnabel, and M. Arcak, “Symmetry reduction for dynamic programming,” *Automatica*, vol. 97, pp. 367–375, Nov. 1, 2018, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2018.08.024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0005109818304230>.
- [11] J. Maidens and M. Arcak, “Exploiting symmetry for discrete-time reachability computations,” *IEEE Control Systems Letters*, vol. 2, no. 2, pp. 213–217, Apr. 2018, ISSN: 2475-1456. DOI: 10.1109/LCSYS.2018.2800125. [Online]. Available: <https://ieeexplore.ieee.org/document/8277172>.
- [12] H. Sibai, N. Mokhlesi, C. Fan, and S. Mitra, “Multi-agent safety verification using symmetry transformations,” in *Tools and Algorithms for the Construction and Analysis of Systems*, A. Biere and D. Parker, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 173–190, ISBN: 978-3-030-45190-5. DOI: 10.1007/978-3-030-45190-5_10.
- [13] H. Sibai, N. Mokhlesi, and S. Mitra, “Using symmetry transformations in equivariant dynamical systems for their safety verification,” in *Automated Technology for Verification and Analysis*, Y.-F. Chen, C.-H. Cheng, and J. Esparza, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 98–114, ISBN: 978-3-030-31784-3. DOI: 10.1007/978-3-030-31784-3_6.
- [14] B. Ravindran and A. G. Barto, “Model minimization in hierarchical reinforcement learning,” in *Abstraction, Reformulation, and Approximation*, S. Koenig and R. C. Holte, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2002, pp. 196–211, ISBN: 978-3-540-45622-3. DOI: 10.1007/3-540-45622-8_15.
- [15] M. Zinkevich and T. R. Balch, “Symmetry in markov decision processes and its implications for single agent and multiagent learning,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML ’01, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jun. 28, 2001, p. 632, ISBN: 978-1-55860-778-1.
- [16] E. Cartan, *La théorie des groupes finis et continus et la géométrie différentielle: traitées par la méthode du repère mobile* (Cahiers scientifiques). Gauthier-Villars, 1937. [Online]. Available: <https://books.google.com/books?id=fELvAAAAAAAJ>.
- [17] E. Leurent, *An environment for autonomous driving decision-making*, <https://github.com/eleurent/highway-env>, 2018.
- [18] G. Brockman, V. Cheung, L. Pettersson, *et al.*, *OpenAI Gym*, 2016. arXiv: 1606.01540 [cs.LG].
- [19] T. Seno and M. Imai, “D3rlpy: An offline deep reinforcement learning library,” *Journal of Machine Learning Research*, vol. 23, no. 315, pp. 1–20, 2022. [Online]. Available: <http://jmlr.org/papers/v23/22-0017.html>.
- [20] P. Polack, F. Althé, B. d’Andréa-Novet, and A. de La Fortelle, “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?” In *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812–818. DOI: 10.1109/IVS.2017.7995816.